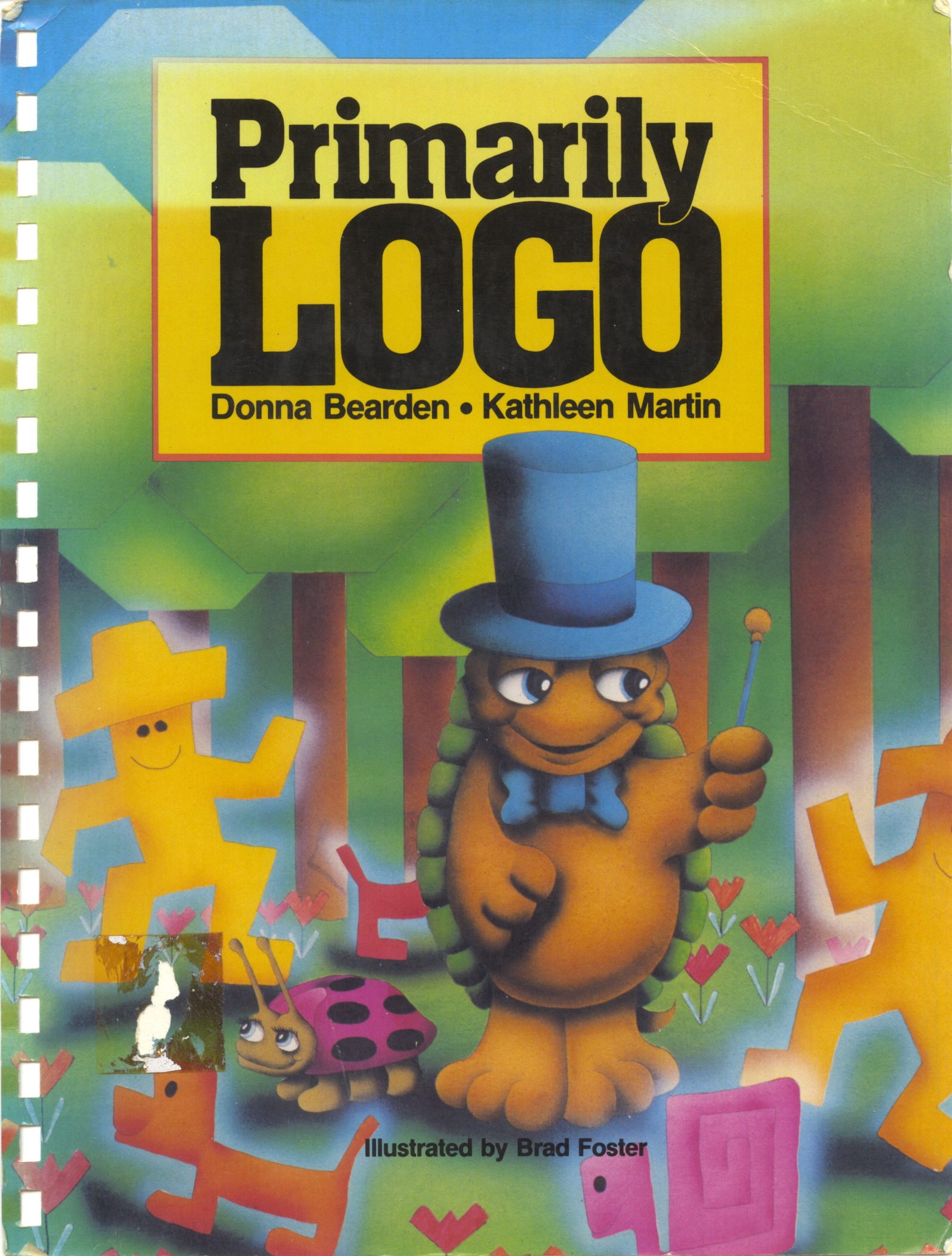
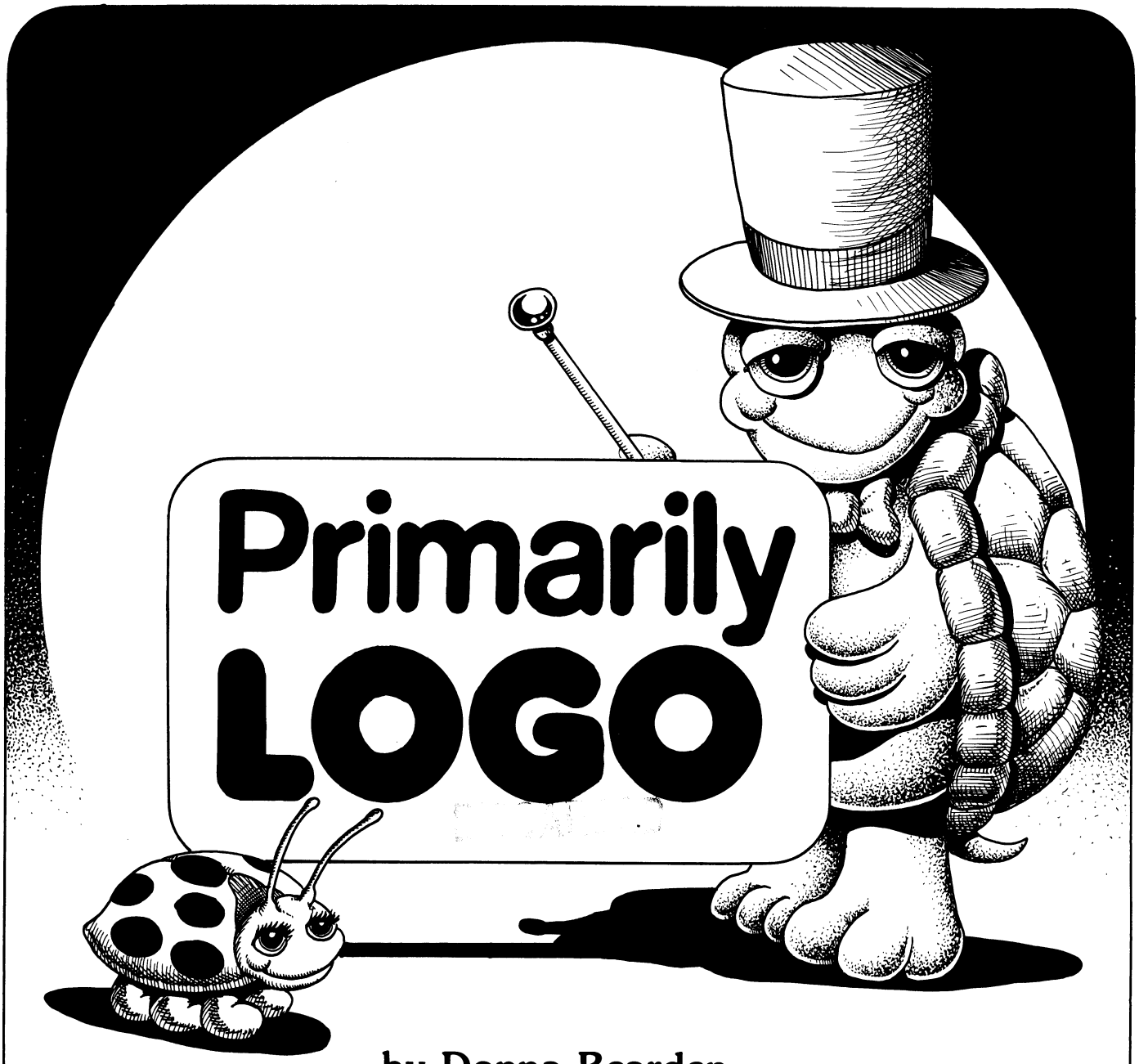
Primarily LOGO

Donna Bearden • Kathleen Martin



Illustrated by Brad Foster



by Donna Bearden
and
Kathleen Martin

Illustrations by
Brad W. Foster

Reston Publishing Company, Inc.
Reston, Virginia
A Prentice-Hall Company

Library of Congress Cataloging in Publication Data

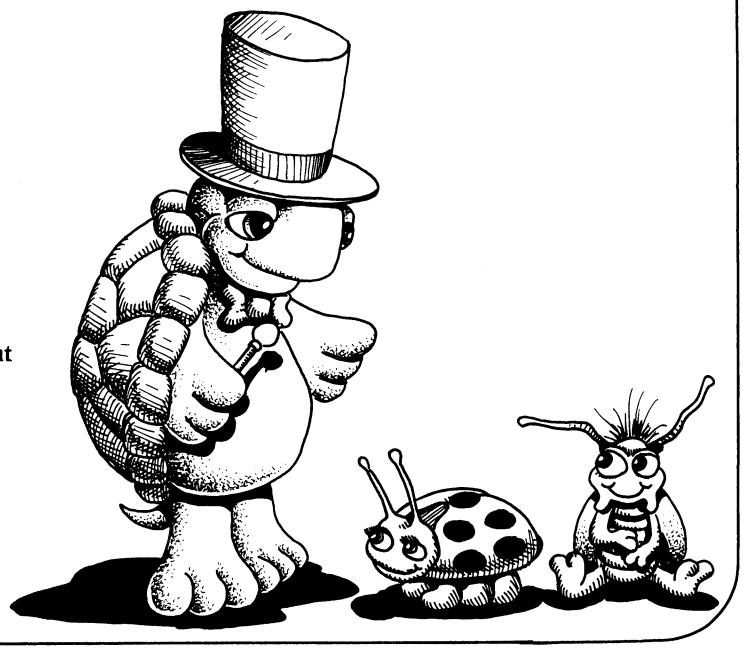
Bearden, Donna.
Primarily LOGO.

Bibliography: p.
1. LOGO (Computer program language)--Juvenile literature. I. Martin, Kathleen. II. Title.
III. Title: Primarily L.O.G.O.
QA76.73.L63B43 1984 001.64'24 84-18142
ISBN 0-8359-5677-6

Published 1984 by Reston Publishing Company, Inc.
A Prentice-Hall Company
Reston, Virginia 22090

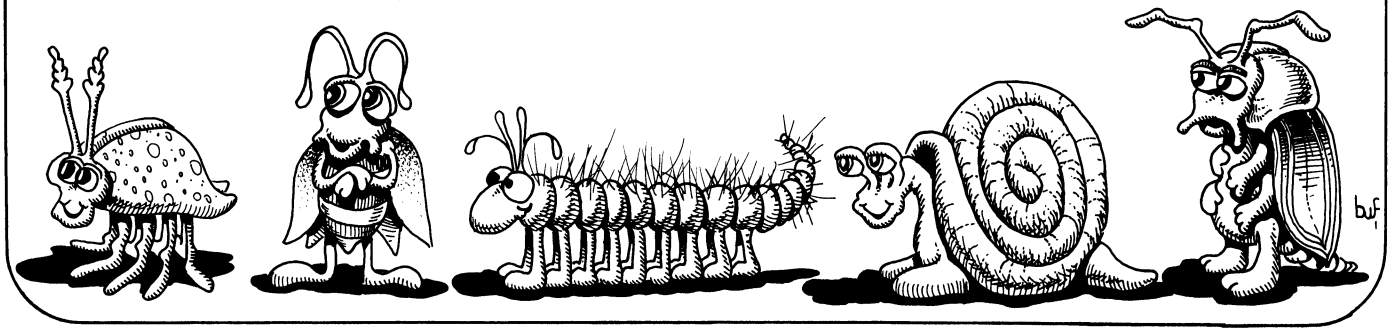
10 9 8 7 6 5 4 3 2 1

Printed in the United States of America



Acknowledgements

We gratefully acknowledge the contributions of Andrew Berner, mathematics professor, who unstintingly shared his ideas with us and whose criticisms of this work were invaluable. His ideas were, in turn, supported by his wife Barbara and son Josh. We also acknowledge the wonderful cooperation we received from our own children, especially Stacy and Scott, who never tired of being our "experimenters." Dan Watt and Harold Abelson are the other adults whom we wish to thank for their beautifully constructed guide books to Logo. Without their leadership we could not have followed.




©brad w foster 1984


©brad w foster 1984


YOU'RE PROBABLY ALL WONDERING WHY I CALLED YOU HERE TODAY...
©brad w foster 1984

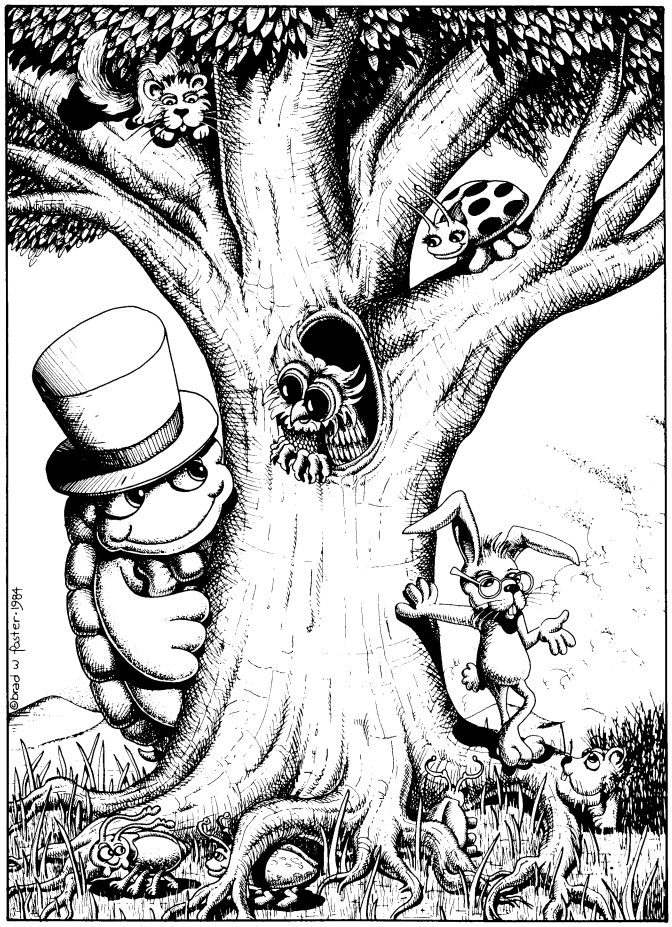
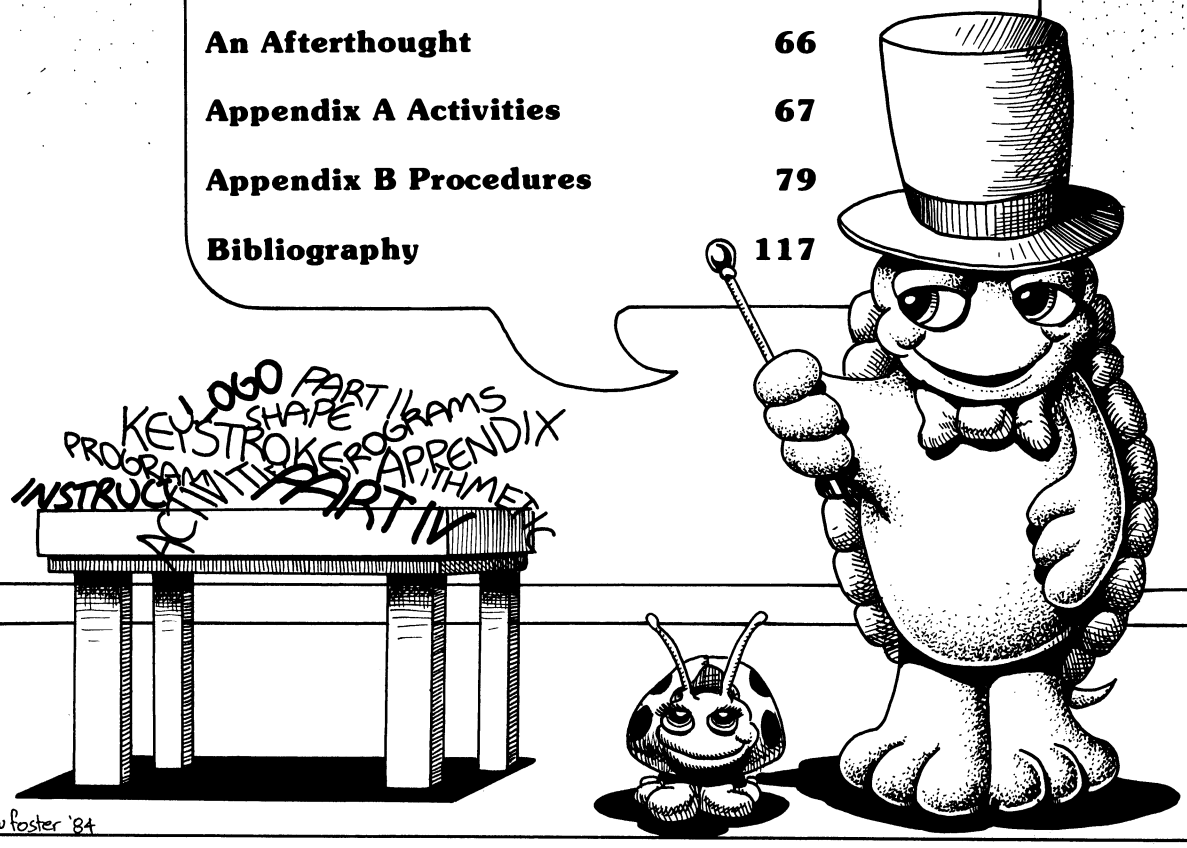
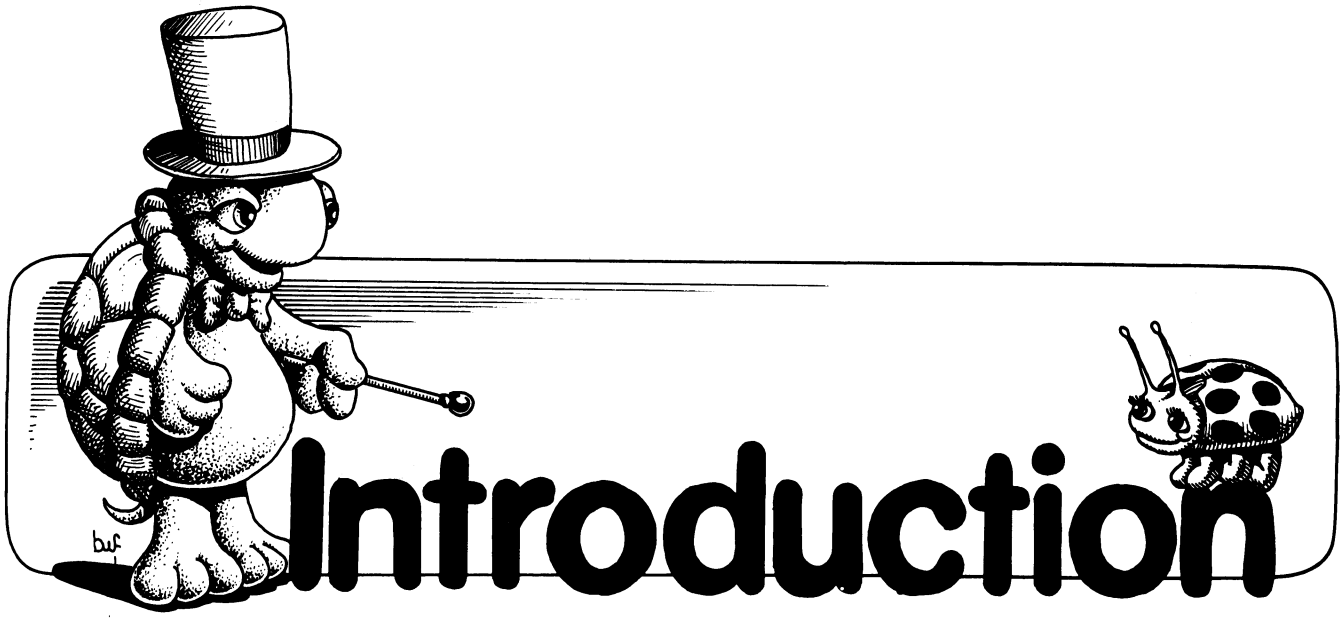

©brad w foster 1984

Table of Contents





Introduction

PRIMARILY LOGO is an effort to cultivate further excitement about Logo among teachers and parents of younger children (5 to 9 year-olds). This is not a book to teach Logo — there are several excellent ones on the market. (See Bibliography for a few.) This is a look at ways to use Logo to explore important concepts with children.*

Part I — Pre-Logo Activities

These take advantage of the child's intuitive sense of body geometry and of some of the toys and games popular at that age level. A single keystroke version of Logo allows children to explore distances and angles and to combine these movements into simple procedures. Several drawing activities are included.

Part II — Beyond Single Keystrokes

The transition to full graphics commands is eased by exploring designs made from a series of single keystroke shapes. Then traditional geometric shapes are introduced. Children step out shapes using footprint patterns before investigating them on the computer.

Part III — A First Look at List Processing

Children construct brags, mad-libs, and a simple adventure game. In the process of learning about variables, they discover an important feature of how computers work.

Part IV — Instructional Tasks

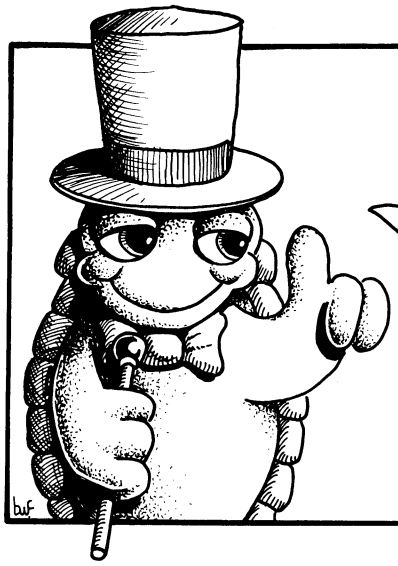
This section offers some ways that a new educational tool might be effectively integrated into a traditional curriculum. Logo programs address concepts of spatial relationships, colors, and simple arithmetic operations. Model programs may be modified to cover other concepts.

A word about our characters.

Toulouse and Lady B. were designed for young children. Teachers are welcome to use them on bulletin boards. For any other use, please contact author/artist first.

*The concepts presented will work with any version of Logo. We have included the commands for both LCSI Logos (Apple Logo, Atari Logo, and IBM Logo) and MIT Logo (Terrepin Logo, Krell Logo and Commodore 64 Logo.) LCSI Logo is used throughout the text. Differences for MIT procedures are noted in italics. Logo-specific disks may be ordered from the authors. See ordering information in Appendix B.


©brad w. foster-1984



Part I: Activities for Pre-readers

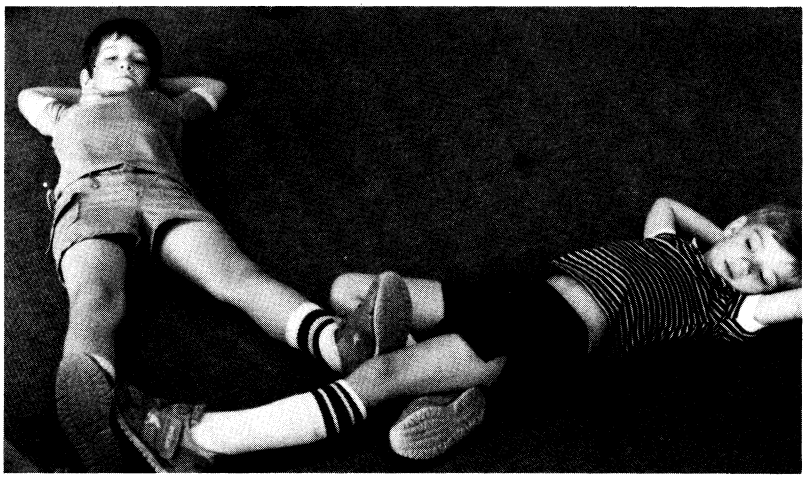
Before children are introduced to Logo, there are numerous activities they can do to become acquainted with the commands and movements of the Turtle. In addition to promoting the same problem solving skills as Logo, these off-computer activities give children an opportunity to experience some abstract concepts in a very concrete way. Possible activities are described here, some of which will be referred to and expanded as we work through this book.

Let's start with two activities that require no additional equipment: Body Geometry and Playing Turtle.

Body Geometry

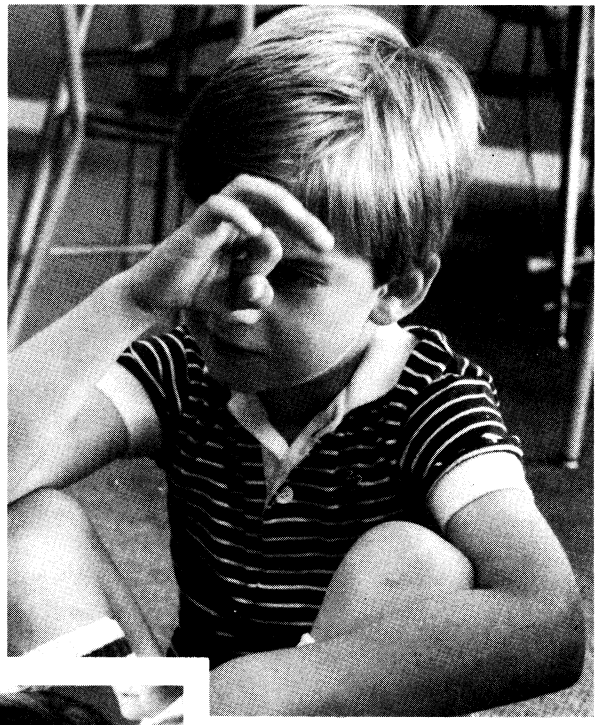
Body geometry is a term for using our bodies to form geometric shapes. It can include simple constructions such as one child forming a circle with his arms or complex structures such as six children forming a pyramid.

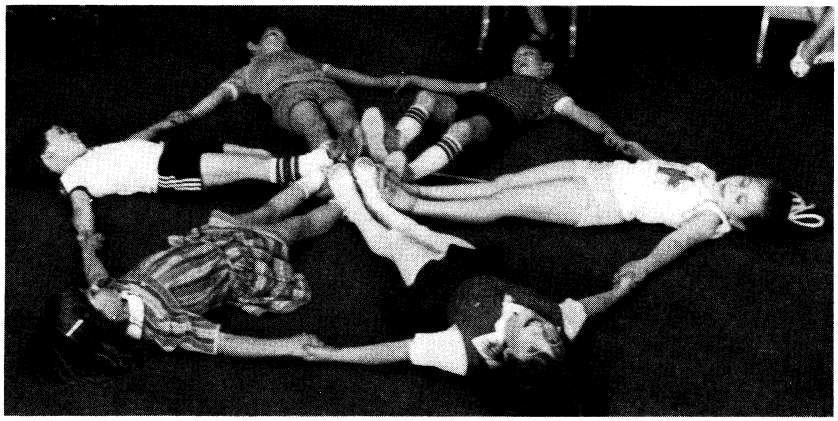
Ask your students how many children it takes to make a triangle. If they answer three, have three children get up and form a triangle. Divide the class into groups of three and challenge them to see how many different ways they can make a triangle. How can two children make a triangle? How can one child make a triangle? With three children forming a triangle, have a fourth child walk around it and describe his or her steps and turns. How many steps to a side? How many turns does it take to end up facing exactly as you started?



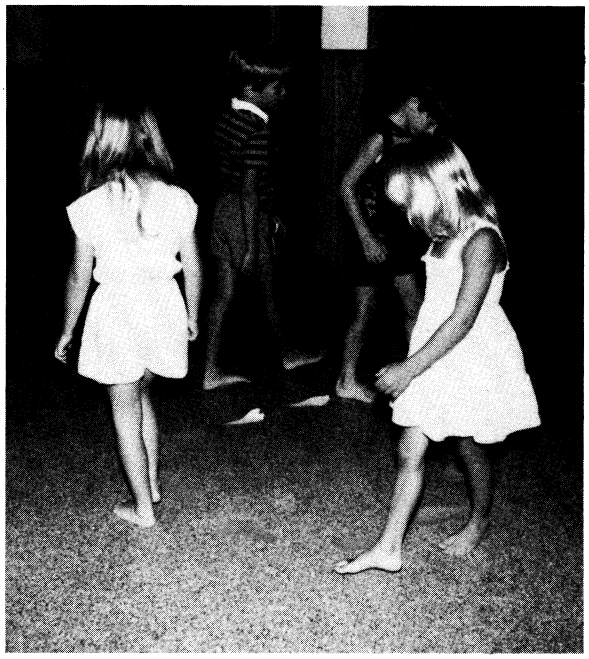
Do the same things with a square.

How about a circle? How many children does it take to make a circle? Ask each child to make a circle. Have two children make a circle, then three, then four, etc., until the whole class is forming a circle.





Talk about the various sizes of circles. What happened as more children joined the circle? Can the whole class make a small circle? Have one child walk around a large circle and describe his or her steps and turns. Have the same child walk around a small circle. Count the steps and describe the turns. It will take fewer steps to walk the smaller circle, but the turns will be greater between each step. Be sure to use a very small circle and a very large one so the differences can be more clearly experienced.



Playing Turtle

As the children form geometric shapes with their bodies, let them command a student, who is the Turtle, to walk around the various shapes.

Send the Turtle out of the room and decide on a location in the room where the Turtle's food is located. When the Turtle comes back, the children will take turns giving him commands to get to his food. The commands will be Forward and Back, followed by a number of steps (baby steps), and Right and Left. Right and Left will mean 90 degree turns. It may be necessary to include commands like "Turn right a little more" or "Turn left just a little." That's ok — that's exploring, just as they will do later on the screen with the Logo Turtle.

Divide the class into pairs and have each partner practice commanding the other. If the children have not learned right and left, use masking tape with R and L on the backs of their hands.

Add a blindfold to the Turtle and place the desks in a maze. The children must command the Turtle through the maze so he doesn't bump into anything.

Big Trak(TM)

Big Trak is a programmable toy tank made by Milton Bradley. The commands it uses are similar to the commands used by the Logo turtle. The only major difference is that the tank's rotation through a circle is based on a clock rather than 360 degrees. In other words, a right angle (or 90 degree turn) on Big Trak is RT 15. If there is a large clock in the room, you can correlate various turns to the different positions of the minute hand. Using Big

Trak makes it possible for children to see commands carried out in a dramatic way. From single commands, children can move to programming a series of commands for the tank to carry out.

One way to introduce Big Trak to a kindergarten class is to have the children sit in a large circle on the floor. After you have explained the commands, have them send the tank from one person to another across the circle. It will take a few times to figure out how far it is across the circle. Have them send the tank to the center, turn and go forward again to get to a child who is not sitting directly across the circle. Make the tank go to the center, turn a complete circle and return to its starting position. Even these simple activities will encourage children to begin estimating distances and angles.

Tape mazes on the floor or set up building blocks and challenge the children to program Big Trak to move through the maze or blocks. Games can be developed in which the tank must pass through certain blocks and knock down others. Can you make the tank dance? Program a series of left and right turns, back and forth, turn around, do the Hokey Pokey! Or how about a huge dot-to-dot on the floor? Cut out 5 or 6 large circles and place them on the floor. Can the children connect the dots with Big Trak? These activities encourage children to think through a series of commands to solve a problem.

You and your students will probably think of several variations to these activities.

Etch-A-Sketch(TM)

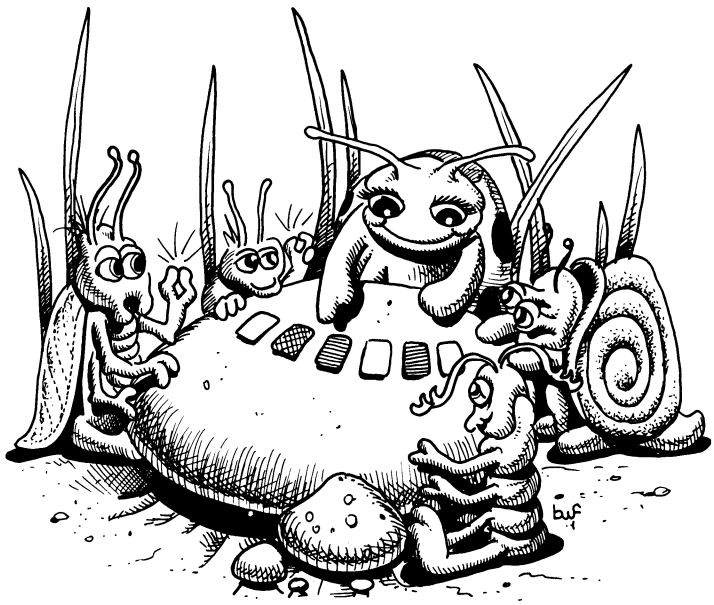
As children draw by turning two separate Etch-A-Sketch dials, they will gain an intuitive feel for the Cartesian coordinate system. Although there are mazes and dot-to-dot overlays available, most are too tedious for kindergarten children. Simply drawing, however, is an activity similar to commanding the Turtle and requires the child to begin thinking in steps. (One kindergarten child even began calling the drawing mechanism the Turtle.) The computer maze patterns included in Appendix A also will fit the Etch-A-Sketch screen.

Pattern Blocks

Through the use of pattern blocks, children can begin to experience relationships between squares, equilateral triangles and hexagons as they form complex designs from simple shapes. They also begin to work with pattern recognition and symmetry. If your class does not have pattern blocks, cut regular polygons from construction paper. (See patterns in Appendix A.) There are several excellent books on the market with pattern block activities. Some are included in the bibliography.

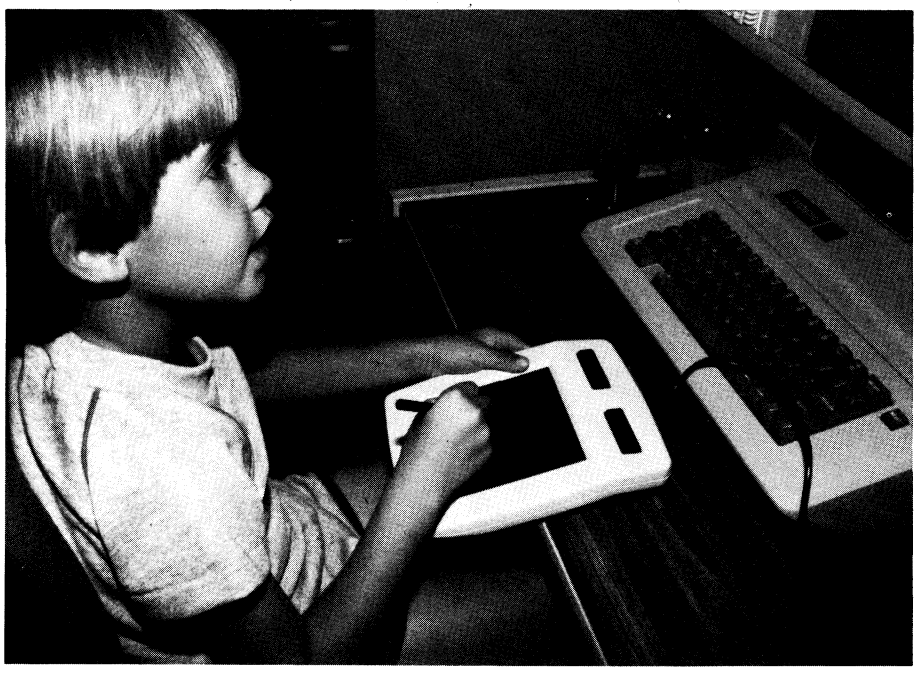
Rhythm Games

So much of mathematics is the recognition of repeating patterns. Thus, even in rhythm games and songs, we can promote the development of pattern recognition. One such game, in which children design and control the pattern is "Snap-Clap-Slap." Cut out several squares of three different colors of paper or felt. Have one child form a repeating pattern with the squares. (i.e., red, red, blue, yellow; red, red, blue, yellow; etc.) One color means snap your fingers, another means clap your hands and the third means slap your legs. As the child points to each square in the pattern, the rest of the group carries out the rhythm. (i.e., snap, snap, clap, slap; snap, snap, clap, slap.)



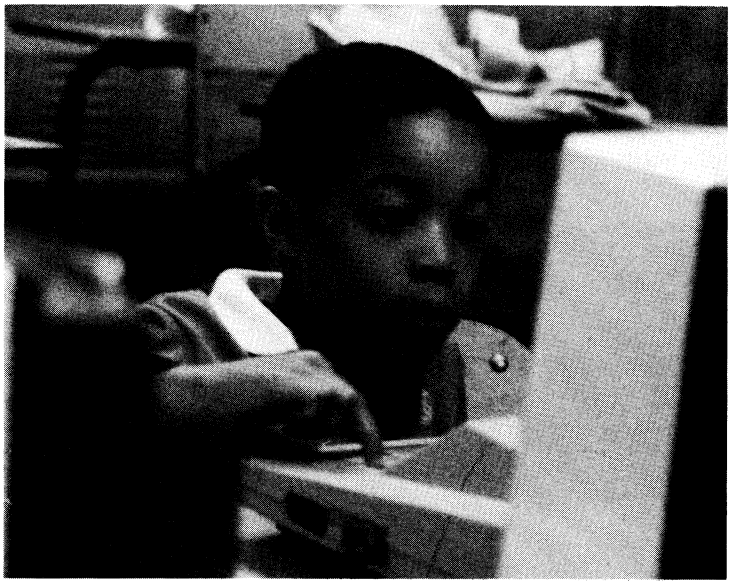
Koala Pad(TM)

The Koala Pad is a drawing tablet that can be connected to the computer. "Drawing" on the tablet creates designs on the screen. Various choices include drawing with straight lines, circles, boxes or freehand. Enclosed spaces may be filled with colors or patterns. Children can doodle or carefully plan their creations. Hand-eye coordination is strengthened as they experiment with lines and colors and move back and forth between the drawing mode and the menu.



Single Keystroke Programs

There are several single keystroke programs already available or listed in various Logo books. You can develop your own single keystroke program quite easily, however, and then add to it and change it as your children progress.



In some single keystroke programs, the Turtle will move immediately when a key is touched. These are called "instant" programs. We will be using them in a variety of activities in this book. The following procedures will require the child to touch a key and then press RETURN or ENTER. Introducing the concept that each command must be entered may make the transition to the full Logo easier.

Look at the following procedures. Notice that we will begin with right angle turns just as we did when we played Turtle. Later we will change the right turn to a 45-degree turn and the children will be an integral part of making the change in the program. They will experience the control over the program and also will experience the greater flexibility that results from the change.

```
TO F
FD 20
END

TO R
RT 90
END

TO L
LT 90
END

TO C
CLEARSCREEN
END
```

(Some versions of Logo will not accept F as the name of a procedure. In those cases, use M for Move.)

These four procedures will allow a child to begin exploring with the Turtle. While you might be tempted to put colored stickers on the keys the children will use, try it first without them. Even very young children will soon recognize the letters they need: R for right, L for left, etc. Allow ample time for just exploring, and remember there's no right or wrong way to explore. Some children will exuberantly command the Turtle all over the screen. Others will think out each move. Some will doodle. Others will immediately set a task for themselves. Attention spans are quite varied. Some will work for five minutes, others for 15 or 30. Be as flexible as you can.

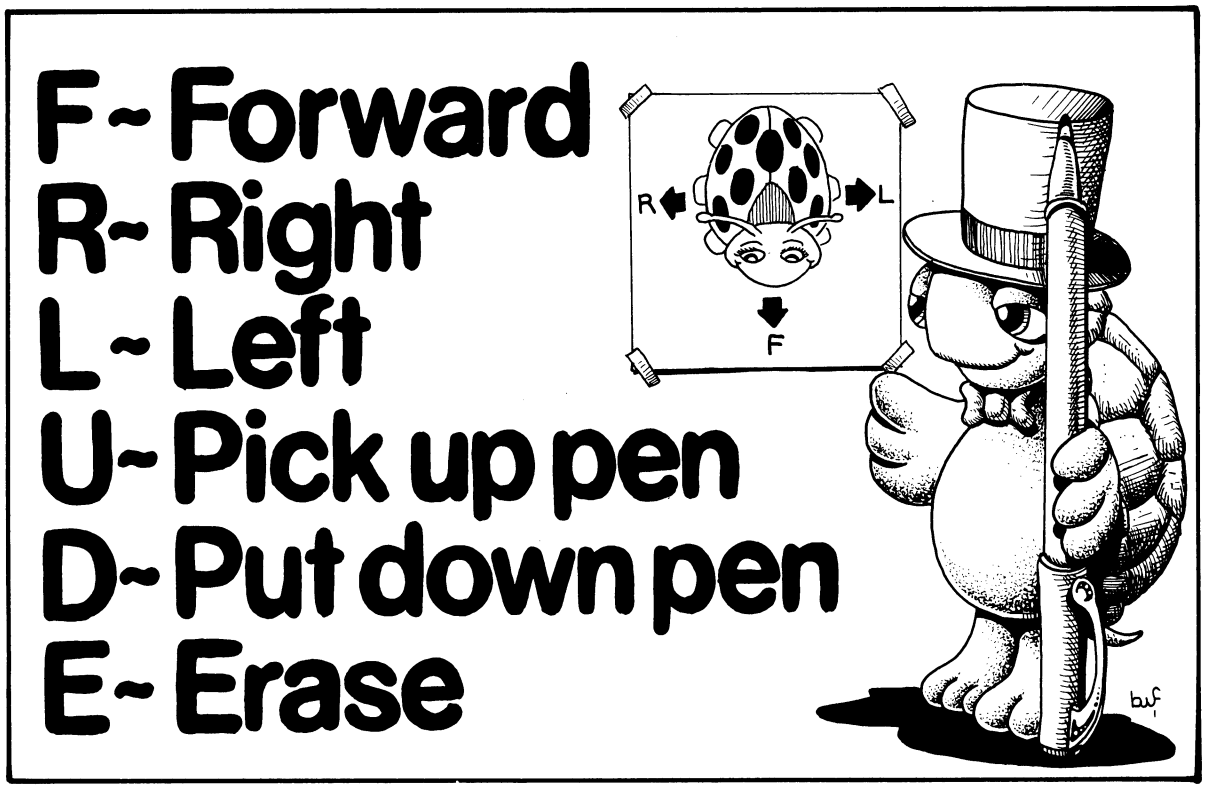
A word of caution. Some single keystroke programs have geometric shapes already defined within them and you may have been tempted to include some in your program. If you do so, you will take away an opportunity for learning and the excitement that comes with discovering something for oneself. For example, think about what is involved in defining a square. Since the turns are already defined as 90 degrees, it will not be difficult for children to draw a square with single keystrokes. When they learn to use the editor, they can learn to define their own square, calling it S for square or B for box or whatever. "Square" will mean so much more if they have had a part in defining it. The computer didn't know how to make a square — they taught it.

At some point, a child will want more information. "How can I erase one line?" "But I want the Turtle to start down here." Once one child brings up the questions, the others will want the information also.

When they indicate a need for additional commands, it's time to add them. Add PENUP, PENDOWN and PENERASE commands. Allow the children to watch as you define these new commands. They may not understand what you are doing, but they will begin to see that procedures can be added and/or changed. Things don't magically happen; we cause them to happen by what we tell the computer.

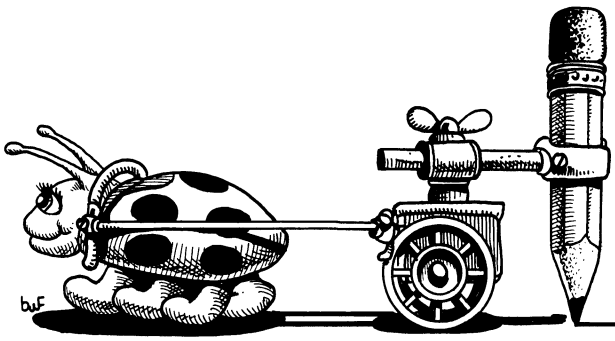
```
TO U
PENUP
END

TO D
PENDOWN
END

TO E
PENERASE
END
```

Each time new commands are introduced, allow time for exploring with them. The child now knows seven commands. Encourage the use of those commands by setting simple tasks. Some will decide on their own projects. When they do, the best thing you can do is observe quietly. For those who need tasks set for them, screen overlays with simple mazes or dot-to-dots are fun. Create your own from clear plastic. You might even put barriers in the maze that will require the Turtle to pick up his pen and then put it back down. In Appendix A are some patterns of simple mazes and dot-to-dots. Since these can be tedious for some children, you might assign them as group projects.



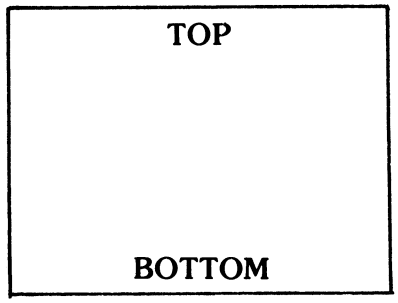
Level 1 Activities

Graph Paper Pictures

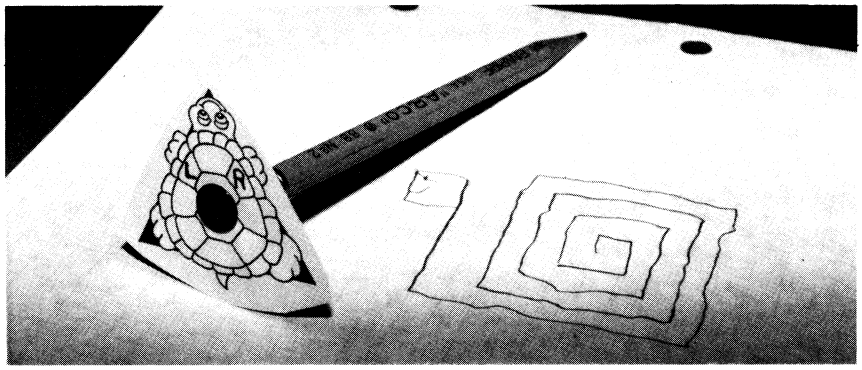


Always combine the work at the computer with related off-computer activities. Children need to experience concepts in a variety of ways. Since we are moving towards creating a design by giving the Turtle a series of commands, let's try a similar activity with graph paper. The children will listen to and follow directions to draw a simple picture with 90 degree turns.

Graph paper marked with faint blue lines is available at school and office supply stores. The lines need to be faint so pencil marks will show over them. Be sure to select paper with large blocks. Place the paper horizontally and have the children write TOP and BOTTOM as shown below.



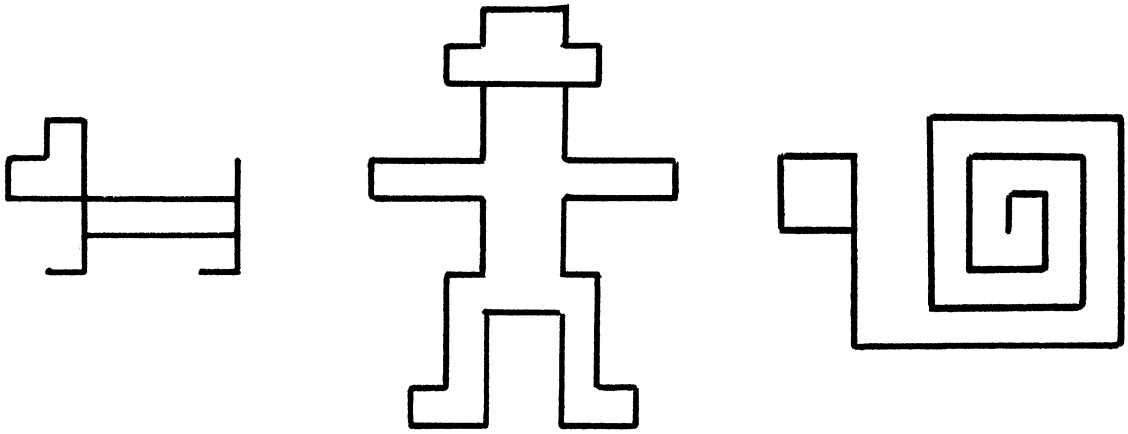
The task is for the children to listen to spoken commands and carry them out with their pencils on the graph paper. It helps the children tremendously if some kind of "pointer" is affixed to the pencil or crayon. Make sure that the "pointer" is firmly attached and cannot slide around on the pencil. (Look in Appendix A for "pointer" patterns.)



The child must turn the "pointer" in the direction given by the command when the command is RIGHT (clockwise) or LEFT (counterclockwise). When the command is FORWARD or BACK, the child moves the number of designated blocks in the direction of the pointer. Have the children start with their pencils in the center of the paper. Each command starts where the last one ended. Emphasize that pencils should remain down, not lifted and returned to the center for each command. It is also possible to have the children move some kind of "pointer" (an arrow or even a small turtle) directly on the paper and then follow it with their pencils. The first time you do this activity, it might be helpful to use an overhead projector. Then the children can see the commands carried out.

Whichever way it is done, this activity may prove difficult for some children because they have to become the "pointer." A lot of twisting around in desks is a good sign that the children are trying to imitate the movement of the "pointer." If, however, the activity is too tedious on graph paper, encourage the children to get up and walk through the commands, leaving some kind of trail. (Masking tape or yarn is probably better than breadcrumbs.)

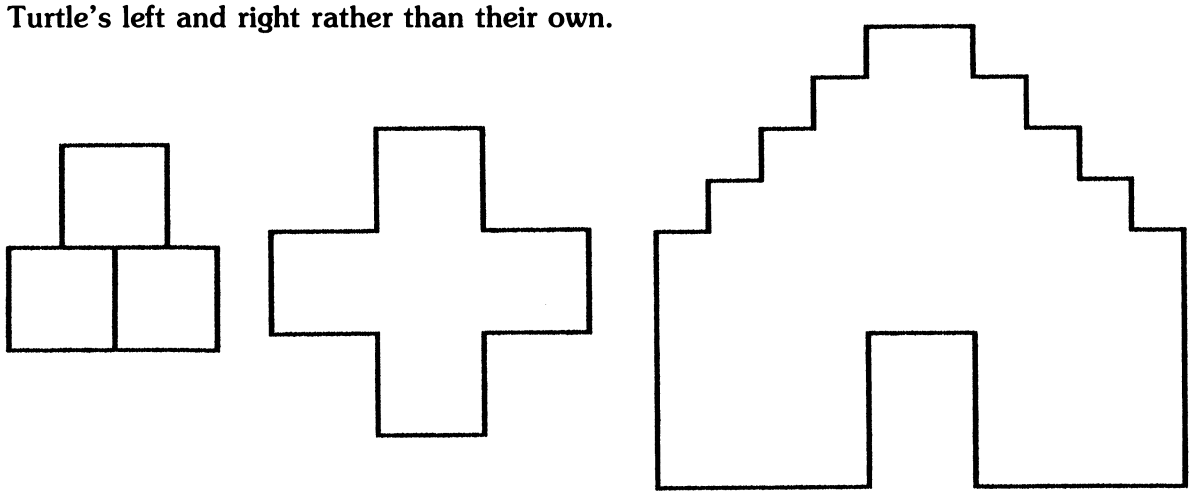
Look in Appendix A for commands to draw three different designs with 90 degree turns.



Have the children draw some of their own designs on the grid paper. Can they figure out the commands for their own designs? Encourage them to name their designs.

Draw.It

DRAW.IT is a computer activity similar to the graph paper pictures. It uses an "instant" program, as mentioned earlier. Load DRAW.IT.LEVEL1 from your disk.* Then type PIC.1, PIC.2 or PIC.3 to start the program. This time the computer will "call out" the commands. A written message appears, such as TYPE F 3 TIMES. After the child types F three times, he types N for next command. If he successfully follows the series of commands, the result will be a simple drawing. Remind the children that the commands will be in relation to the Turtle's left and right rather than their own.



*All procedures are listed in Appendix B. You may make your own **Primarily Logo** procedures disk. If you wish to order a Primarily Logo disk already made, see ordering information on page 80.

Point out to the child that the computer will not check for mistakes. He must work carefully and check himself. If the child makes a mistake and realizes immediately that he has typed one too many F's or turned one R too many, he can correct it. In the case of the turns, he can simply type the opposite turn before he types N to get the next command. If he has gone forward too far, he will have to type E for erase, type B for back and then type D to put the pen back down. (If he has gone back too far, he types E then F then D.) If the child realizes he made a mistake several commands back, teach him how to stop the program completely and start over by retyping the name of the procedure. This simply gives the child additional control over the program and the computer.

Defining Procedures

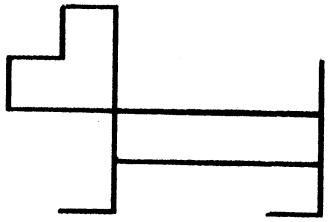
After the children have experimented on the computer for some time and have worked through some of the graph paper pictures and DRAW.IT pictures, they may want to design something of their own that the computer will "remember." Defining procedures can be done quite easily with the single keystroke procedures you have written. Probably the most difficult part of defining a procedure, for kindergarten students, is writing down commands so they can later be transferred to the editor. Have the children work in pairs, taking turns writing down commands for each other. Since we are working in single keystrokes, it will only require writing single letters for each movement of the Turtle.

Some children might find it easier to plan a design ahead of time using graph paper. Others will want to use the screen as a drawing pad. Allow them to select the option that works best for them. Also, if you allow them to choose their own projects, they will be more likely to "own" them. In all likelihood, they will work longer and harder at self-defined projects than assigned ones. Be prepared to help them simplify their ideas if the level of frustration becomes too high.

Once they have worked out the commands to draw their design and have a written record of them, show them how to use the editor. When defining procedures in the immediate mode, corrections cannot be made once RETURN has been pressed. Thus it might eliminate some frustration to teach the students to go to the editor with EDIT "*name*. (This is not a problem with the MIT versions of Logo since typing TO *name* puts you into the editor.)

Here is an example of a stick figure of a dog. The procedure might look like the one below. (Explain to the children that more than one command can be typed on the same line as long as there is a space between each one.)

```
TO DOG
F F L F L F
R F L F L
F F F F F F
L F R R F F F
R F R R F L F
L F F F F L F R
F R R F L F F
END
```

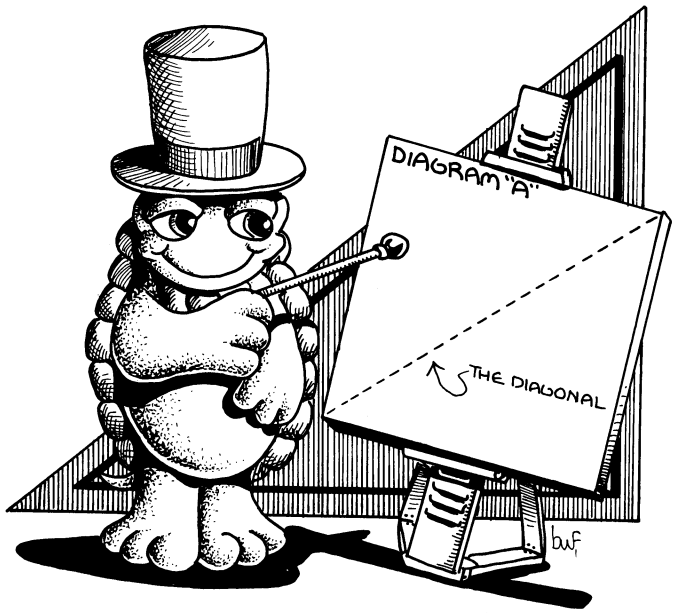


That may seem like a lot of typing. But what seems tedious to us may not be to a 5- or 6-year-old. If they have chosen their own projects, they will work hard to complete them. And once they are completed, what an achievement — new procedures, ones that will cause the Turtle to draw a dog simply by typing DOG, or a cat by typing CAT or a whatever by typing WHATEVER.

Once the children have defined something, have them use U and D to place the design at a particular location on the screen. If it's a small design, have them repeat it several times on the screen.

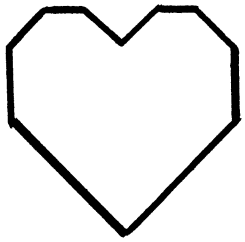
Level 2 Activities

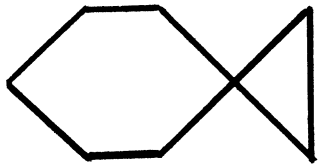
After the children have done a few graph paper pictures and perhaps transferred some of them to the computer, introduce the diagonal. In all likelihood, by now you have had a student who has wanted to use it.

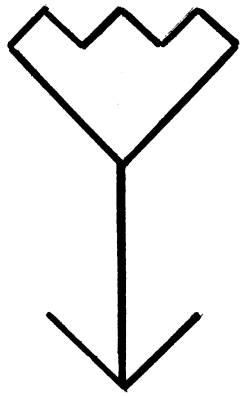


The diagonal will involve splitting the squares on the graph paper. Talk with the children about "half-turns," meaning turns that are one-half as much as the turns that they used before. Tell them to point the "pointer" upwards along a line on the graph paper and then to make one half-turn to the RIGHT. Is the "pointer" aimed at the opposite corner? Now have them move FORWARD. Where do they end up? Does the line split the square into two triangles? Using an overhead projector can be particularly helpful in explaining half-turns to the children.

Included in Appendix A are series of commands to draw several graph paper pictures with diagonals.







Transferring Designs to the Computer

Aha! Now when we want to transfer pictures to the computer, we will have to change the turn the turtle makes from 90 degrees to 45 degrees. Since the children have already experienced the edit mode, it will be a logical step to change a procedure. Have them help you edit the R and L procedures.

```
TO R
RT 45
END

TO L
LT 45
END
```

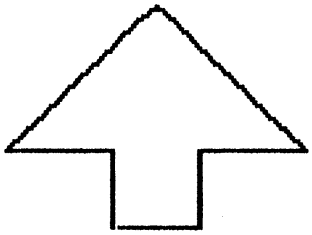
This is a very good demonstration that you and your students are in charge of the computer. If you need more flexibility, you build it in. You change the procedures to fit what you want to do.

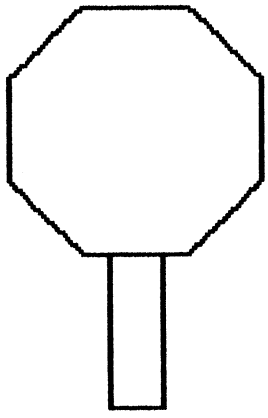
Of course any pictures designed with R and L at Level 1 will not work right once R and L are redefined. You have two choices. The children can edit their procedures to correspond to the new turns or you can save the old procedures as Level 1 procedures with R and L defined as 90 degree turns and save the new procedures as Level 2 procedures with R and L defined as 45 degree turns. If the designs are not too complicated, they will not be difficult to edit. The children will realize that any R or L commands simply have to be changed to two R's or two L's.

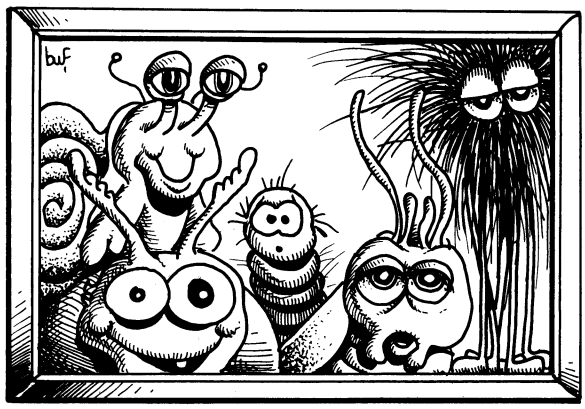
Other Diagonal Activities

Included in Appendix A are mazes and dot-to-dots requiring 45 degree turns.

In DRAW.IT.LEVEL2, the turn is defined as 45 degrees. The FORWARD has been redefined as 10 instead of 20.







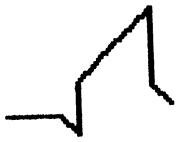
Crazy Critters

Some children (and adults, too) get so intent on producing "something" that this activity is a way to get them back into the "play" mode, exploring with different numbers and turns. It is often when something unexpected happens that the most learning takes place.

To create "crazy critters," have the children go directly to the edit mode and type several commands, perhaps 20 or 30. (Call it Z for craZy.) Then go back to the immediate mode and try Z. Was it what they expected? Type Z and enter it several times and see what happens. By now some children will able to go to the edit mode and visualize and plan what they want the Turtle to do. This is quite abstract, however, for young children. This activity is meant as a fun way to practice using the editor as well as an opportunity to explore the unexpected.

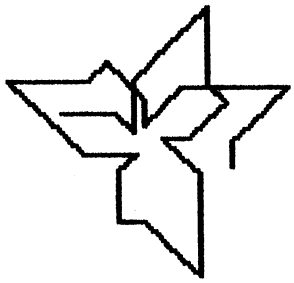
Here's an example of a Z procedure:

```
TO Z
R R F F R F L L L F F R F
F R L F F R R R F F F L F
END
```

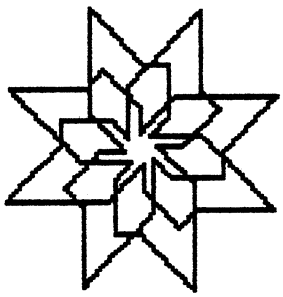


Doesn't look like much, does it?

What if we type Z four times?



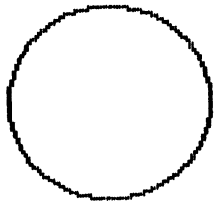
And if we type Z eight times? Say, not bad! And totally unplanned!



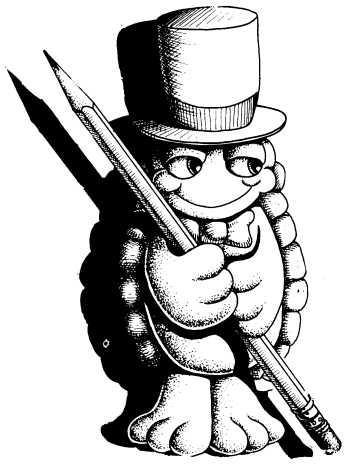
Level 3 Activities

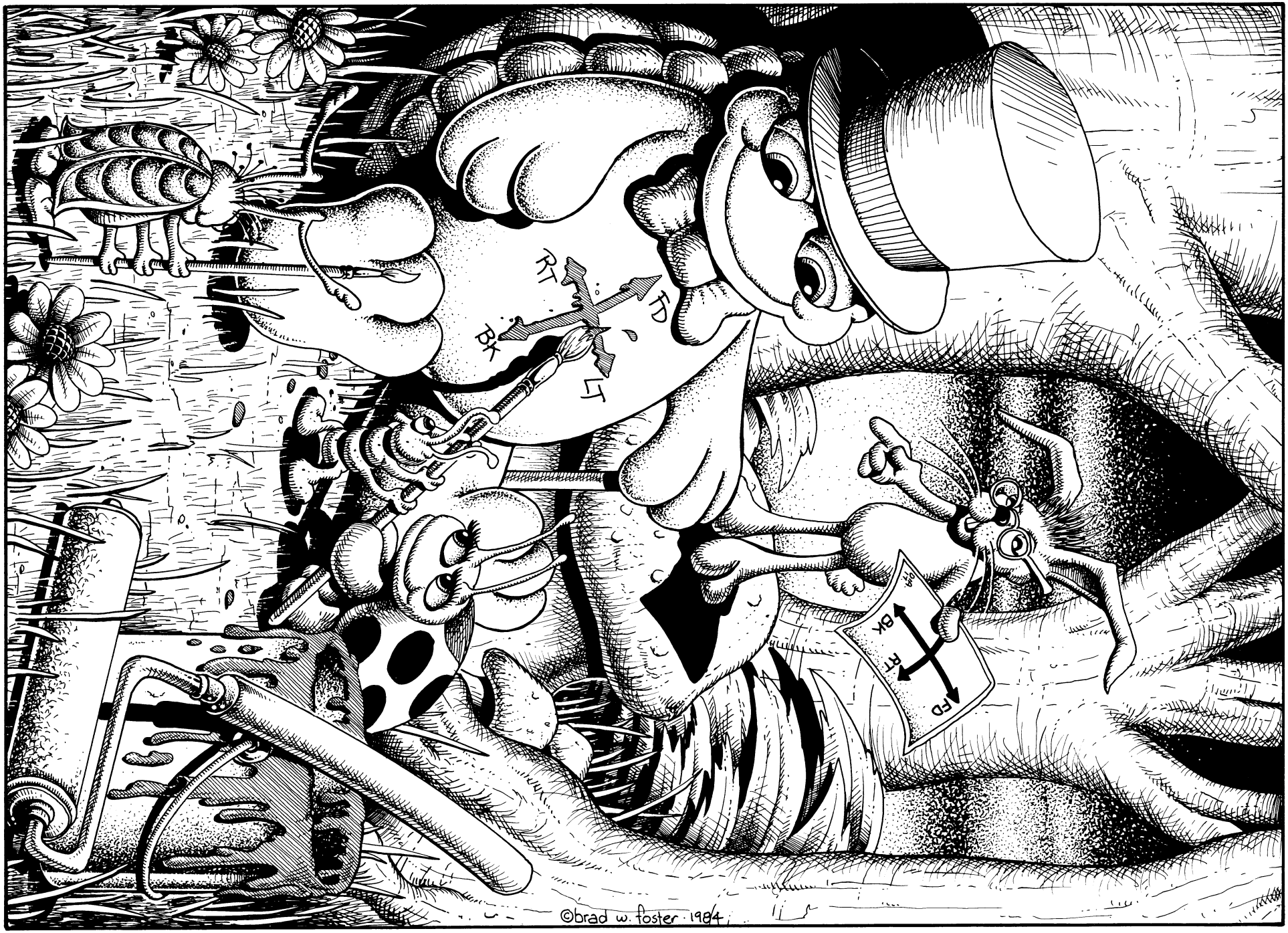
You and your students can continue to expand the single keystroke program, changing the amount the Turtle turns as the children want to refine their drawings. We have included a DRAW.IT.LEVEL3 which has 15 degree turns. As the turns become less, the children can experience the Turtle getting closer and closer to drawing a curve. Do not rush them to get to this point, however. It is important that they understand each step and build on their understanding.

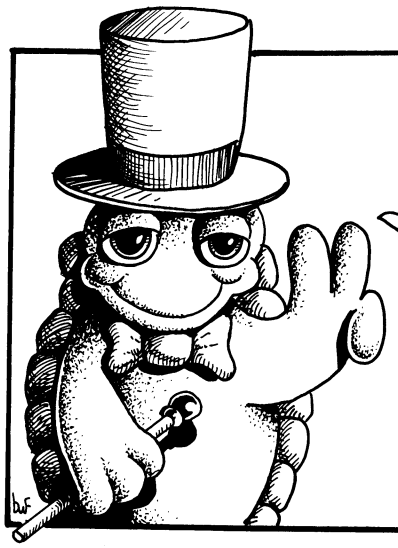
In addition to creating pictures, the single keystroke concept can be carried over into specific curriculum areas. (How about a dot-to-dot of a constellation as you study the stars?) You can use the power of Logo and the ease of single keystrokes to make the computer a teaching tool individualized for your classroom and your students.



NOTES




RT
FD
BK
LT
BK
RT
FD
©brad w. foster 1984



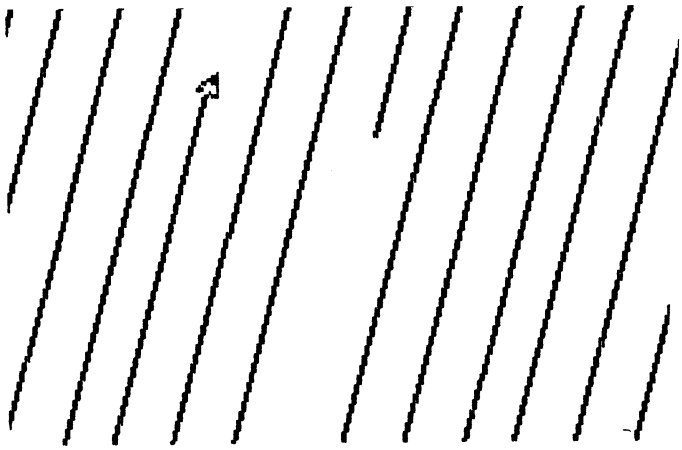
Part II:
Shape-Making Procedures

In Part I, we discussed single keystroke commands and suggested several activities using those commands. The purpose of the single keystroke commands is to introduce younger children to Logo-like experiences. While these experiences are quite narrow compared to what Logo itself affords, they do offer children a manageable beginning. Hopefully, this beginning will develop into a serious and sustained interest in Logo programming as the children mature.

Weaning children from the single keystroke commands is usually not difficult because they naturally begin to want more control over the movements of the Turtle. When introducing the primitive commands, remember not to rush the children. Although they may have been working with a single keystroke version of Logo for a year or more, they will only be familiar with a few angular measures and have limited experience with estimating distances. Allow them time to play around with the commands.

Introduce ST (for SHOWTURTLE) first. The triangular Turtle appears in the center of the screen, or the HOME position. Explain the commands FORWARD (FD), BACK (BK), RIGHT (RT), and LEFT (LT). Stress the fact that these four commands are **always** followed by a space and then a number. The children should already be accustomed to hitting the RETURN key after each command. FD 1 moves the Turtle one turtlestep in the direction that it is pointing. One turtlestep is a very, very short distance. BK 30 moves the Turtle approximately 1 inch backwards from the direction it is pointing. RT 180 turns the Turtle to point opposite its present direction. What does LT 180 do?

Encourage the children to try lots of different numbers with the commands. After they have had ample time to fill the screen with "doodling," introduce CS (for CLEARSCREEN). If they try a large number such as FD 3000, the Turtle will "wrap" around the screen. If the Turtle has been rotated first, the wrap might look like this:

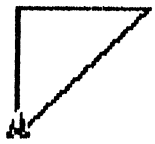


Suggest to the children that they turn the Turtle in another direction and "wrap" again. The child's fascination with large numbers makes the "wrapping" phenomenon a particularly popular activity. It is best to allow the children to explore it fully in order that it not prove distracting later.

Ask the children to type FD 50 and then RETURN. Now type HOME. The Turtle returns to the center of the screen by backing up along the same path. Ask the children to type the sequence of commands below.

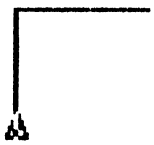
```
FD 50
RT 90
FD 50
```

Now when HOME is typed, the Turtle does not retrace its path, but returns home along the shortest route.



As it returns home, the Turtle leaves a trail. Introduce PU (for PENUP) as a way of preventing pen marks when they are unwanted. Whenever PU is used, PD (for PENDOWN) is necessary to return the Turtle to the drawing mode.

```
FD 50
RT 90
FD 50
PU
HOME
```



Tell the children that there is another command they can use whenever they make a mistake in their drawings. PE (for PENERASE) *(PC 0 for MIT)* makes the Turtle erase whatever lines it moves over. Have the Turtle move FD 50 RT 90 FD 50 one more time. Ask the children to type PE BK 50 LT 90 BK 50 *(PC 0 BK 50 LT 90 BK 50).* What happens to the lines? PE is like PU; PD must be used to return the Turtle to the drawing mode. *(To return to the drawing mode in MIT Logo, use PC 1.)*

Encourage the children to do a project using the commands that they have learned. Some may want to practice their projects on paper first; others may prefer working directly on the screen. One project that children frequently enjoy is drawing their initials. Whatever their projects, it is important that children be given time to practice the commands and to explore some of their own ideas. Invariably, good questions emerge out of these explorations and thereby provide the teacher with rich opportunities for individualized instruction. It is important not to sacrifice these opportunities to too rigid a task orientation. The alternation between teacher-assigned-tasks and student-initiated-projects seems to offer an effective learning environment for Logo.

Single Keystroke Shapes

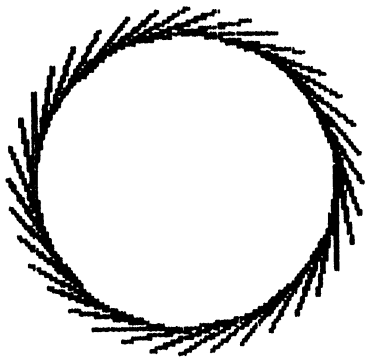
Children usually enjoy developing a single keystroke collection of shapes out of which they can construct more complex designs. The procedures defining a few such shapes are given below.

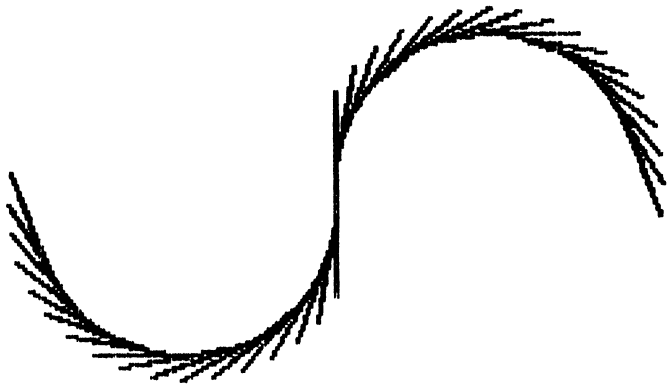
```
TO Y
FD 40
BK 30
RT 10
END
```

```
TO T
FD 40
BK 30
LT 10
END
```

```
TO X
FD 20
BK 20
RT 10
END
```

Begin by having the children define the procedure Y. If they have spent sufficient time with the single keystroke version of Logo, they will be quite familiar with the edit mode. Encourage them to play with Y and discover what they can make using only that key and the procedure it defines. Two likely consequences of their play are the designs below.



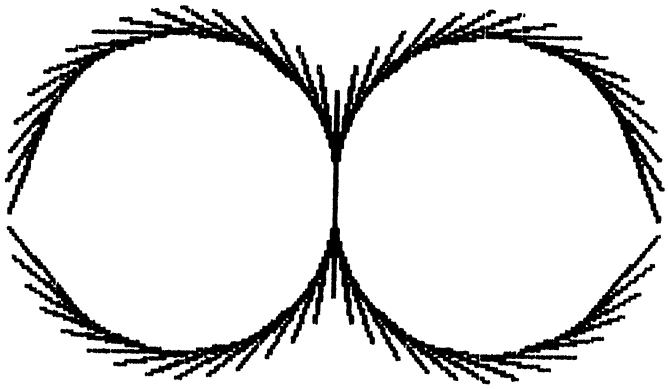


Write procedures for each of the designs, introducing the children to the convenience of the REPEAT command.

```
TO FUZZY
REPEAT 36 [Y]
END
```

```
TO WUZZY
REPEAT 12 [Y]
PU HOME PD
RT 180
REPEAT 12 [Y]
END
```

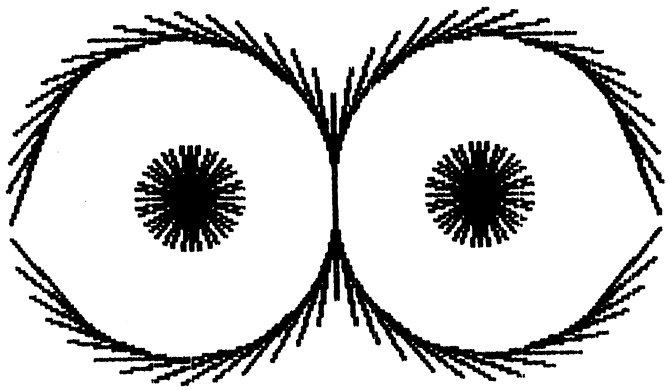
Some child will probably ask for a Y that goes to the left. Write such a procedure with them, calling it T. Now it is possible to transform WUZZY to EYES.



Make sure that the children keep an accurate record of the work that they do on the screen in the immediate mode. This record will enable them to write a procedure for EYES.

```
TO EYES
REPEAT 17 [Y]
PU HOME PD
REPEAT 17 [T]
PU HOME PD
RT 180
REPEAT 15 [Y]
PU HOME PD
RT 180
REPEAT 15 [T]
END
```

Eyes, of course, need eyeballs. The X procedure provides a nice sparkling LOOK.



```
TO LOOK
EYES
PU HOME RT 90 FD 55 PD
EYEBALL
PU RT 180 FD 110 PD
EYEBALL
END

TO EYEBALL
REPEAT 36 [X]
END
```

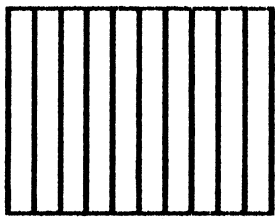
Some children may prefer to do everything in the immediate mode on the screen; others seem to appreciate the efficiency of procedures. Again, it is important not to rush the children. A child who persists in the immediate mode probably needs a greater accumulation of concrete experience. A child who is pushed into writing procedures before he is ready to understand those procedures will frequently return to the immediate mode as soon as the teacher leaves.

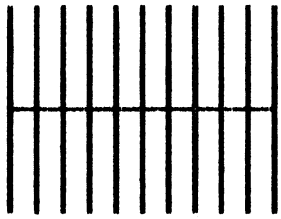
Another shape that is fun to fool around with is W.

```
TO W
FD 40
RT 90
FD 10
RT 90
FD 40
RT 180
END
```

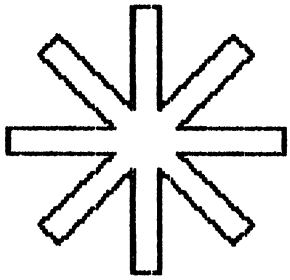
A bunch of Ws in a row looks like a picket fence or a comb.

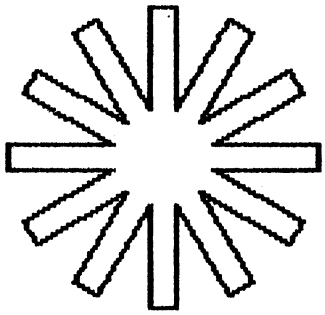
Invite the children to make these designs.



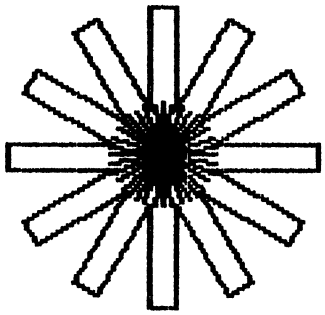


What about trying a turn after each W?

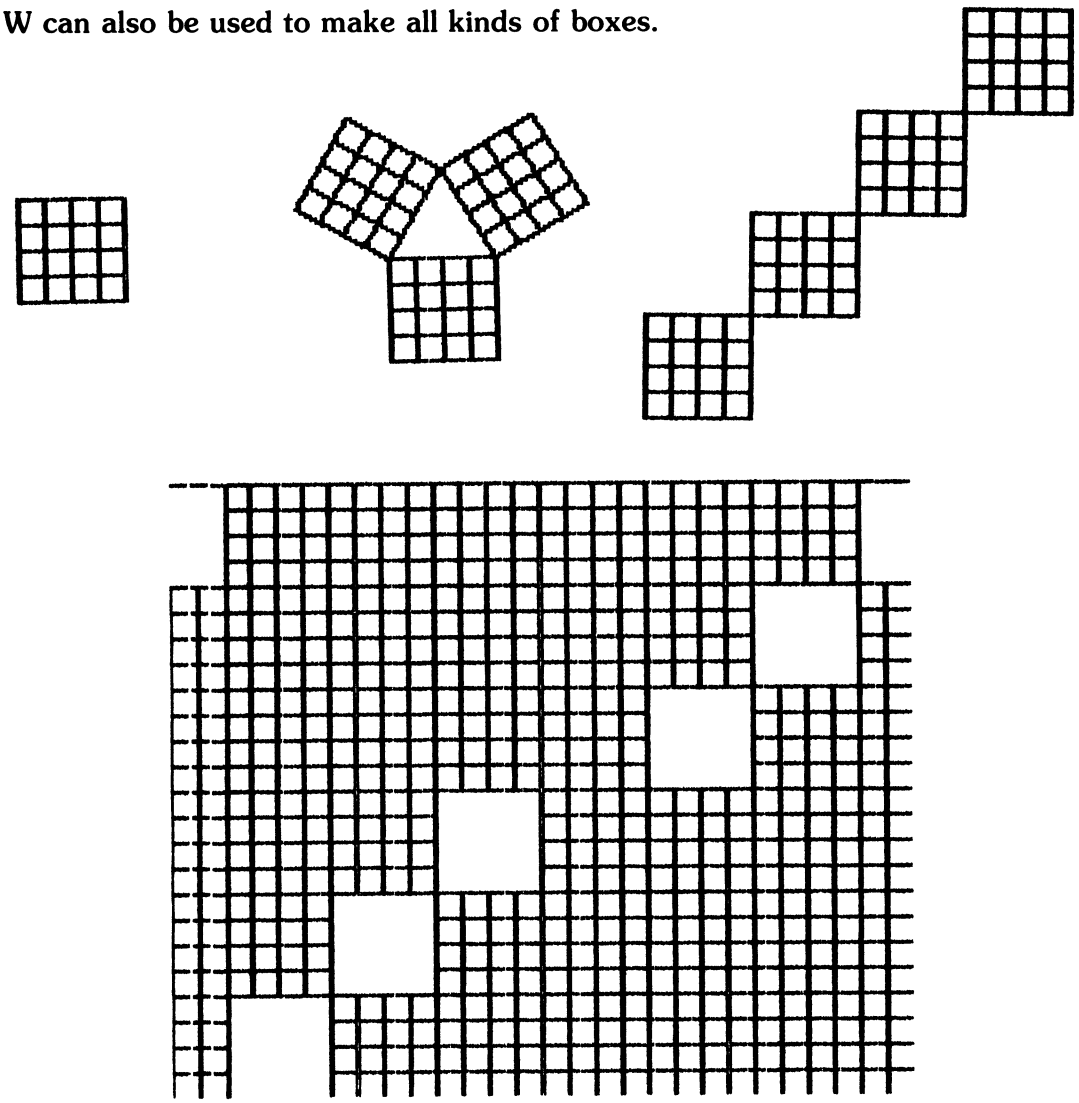




Some child might even decide that the 12-prong W would look better with an eyeball in the middle.



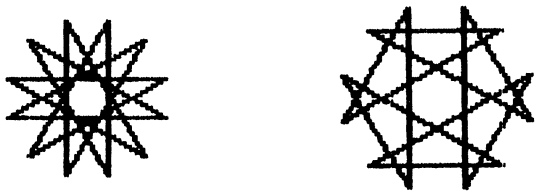
The W can also be used to make all kinds of boxes.



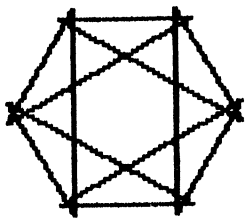
Even a shape as simple as the V can yield some delightful designs.

```
TO V
FD 60
RT 150
FD 60
RT 150
END
```

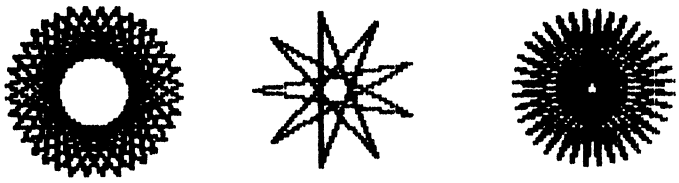
Eight Vs make a star, while slipping a FD 20 between each V produces an attractive wire-like design.



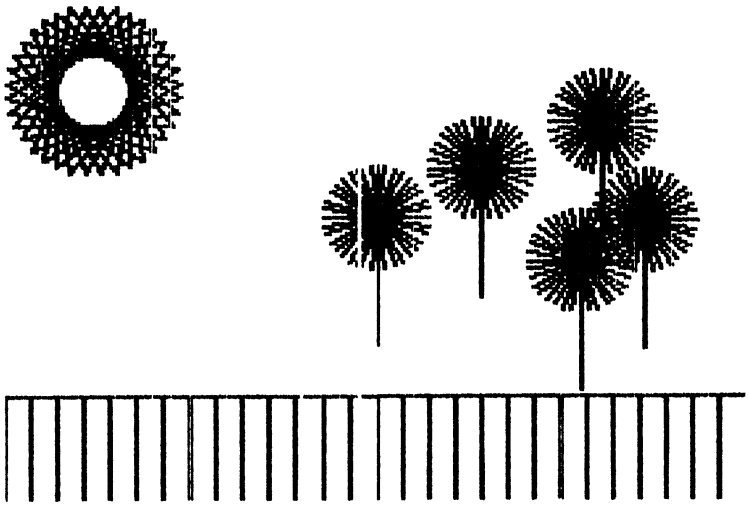
FD 30 between each V makes a star within a hexagon.



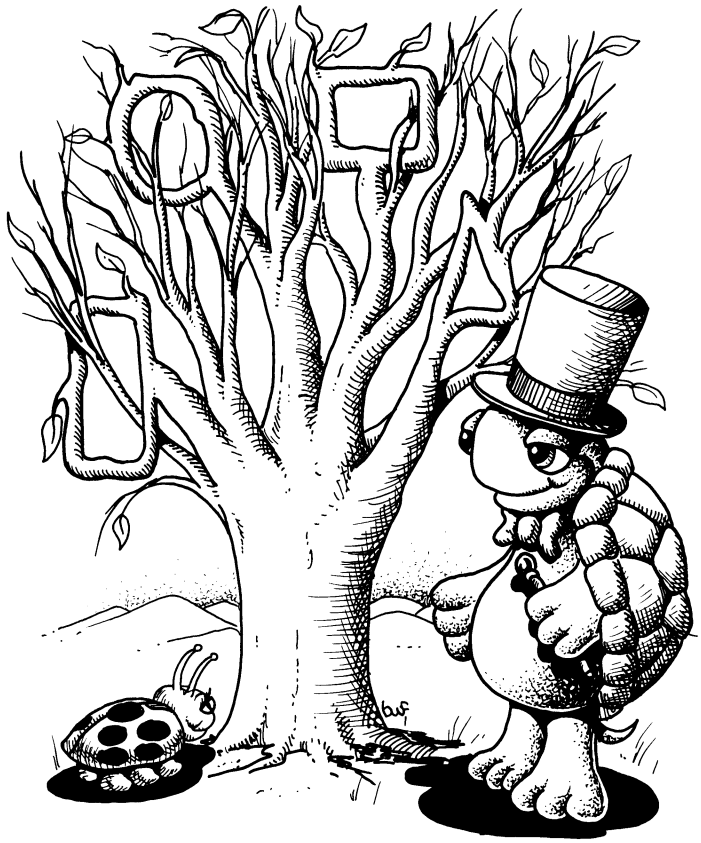
The other star-like designs below result from changing the angle value of V and then simply repeating the V.



If the children continue to show interest in the shape designs, encourage them to write procedures for their own shapes or to make a design using lots of different shapes. In the design below, the sun (modified Vs) is shining on a bunch of flowers (Xs) behind a picket fence (Ws).



Traditional Geometric Shapes



In exploring their single keystroke shapes, the children probably discovered some of the traditional geometric shapes such as circles, squares, and triangles. These shapes can be better understood if children engage in a number of activities off the computer. If the children have not yet tried the body geometry described under the pre-Logo activities, it would be well to start there. Then try an activity using the children's own footprints.

Give each child two sheets of paper. Ask them to trace each foot on a separate piece of paper and to cut them out. Collect the cut-outs. Ask all of the children except two to form a circle by linking arms. When the circle is complete, ask one of the children to walk around the circle taking babysteps. The second child should put a foot cut-out at each step taken by the first child. The result should be a circle of footprints. Ask the children to step back and to describe the movements of the first child as shown by the footprints. Tell the children that they should describe the movements in turtle commands.

Leave the circle of footprints on the floor and have the children go to the computer to make a circle. Instruct them to keep an accurate record of all commands. If they apply what they learned off the computer, they will probably use commands that send the Turtle FD 1 and RT 1 (or LT 1). If they do not already know, show them that they can write the commands on the same line: FD 1 RT 1. If they persist in the tediousness of one command at a time, remind them of the REPEAT command. Be sure that the children continue to record all commands. When the circle is finished, they should total the number of turns. A turn of 1 (RT or LT) should require a total of 360 repetitions. The procedure can then be simplified. If the circle goes off the screen, use PENUP (PU) to reposition the Turtle prior to circling.

```
TO CIRCLE
REPEAT 360 [FD 1 RT 1]
END
```

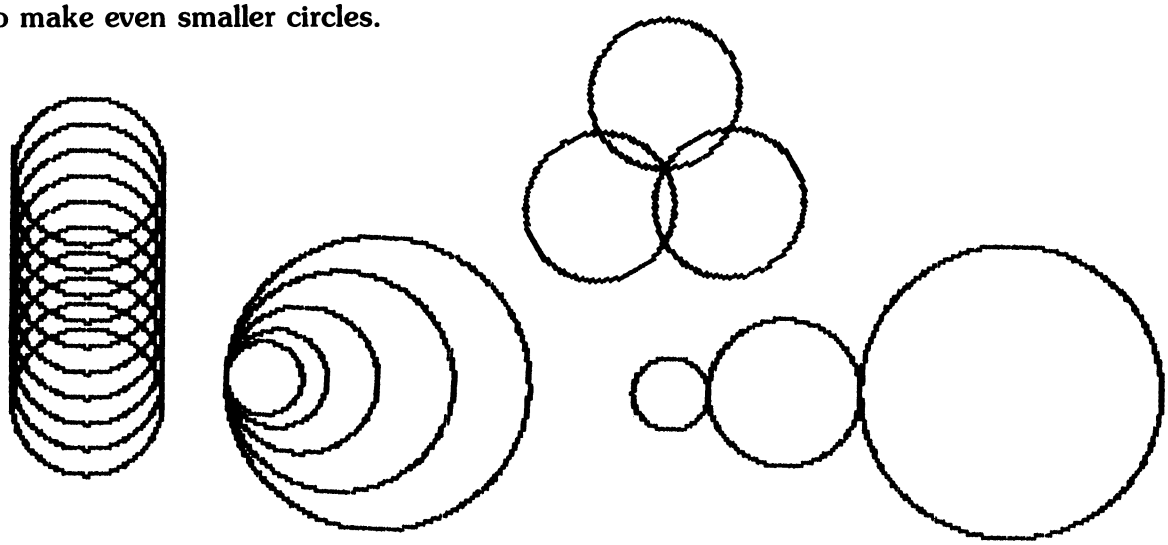
The children will probably want to make smaller circles. Before trying it on the computer, have them repeat the off-computer circling activity. This time make a smaller circle by using only half of the children. When the children compare the first circle of footprints with the second, what do they notice? They will probably see that there are fewer footprints. Have them look again. Encourage them to pay particular attention to the amount of turn of the footprints. How do they compare in the two circles? They should notice that the footprints in the smaller circle are turned more.

Instruct the children to go back to the computer and write a procedure to draw a smaller circle. If the turn is changed to RT 2 (or LT 2), what happens to the circle? The children will probably write the procedure as follows.

```
TO CIRCLE2
REPEAT 360 [FD 1 RT 2]
END
```

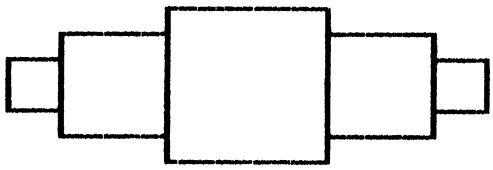
When they execute the procedure, they will notice that the turtle goes around the circle twice. Talk with them about how they can change the procedure so that the turtle will only make one trip around. Through the discussion they can be led to understand that they need to REPEAT only 180 times rather than 360.

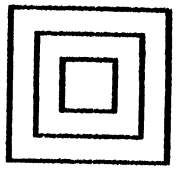
Suggest to the children a number of different circular designs that they might want to try. Encourage them to make their own designs. If they wish, they can write procedures to make even smaller circles.

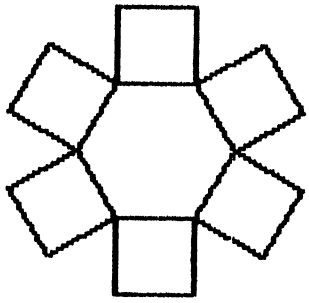


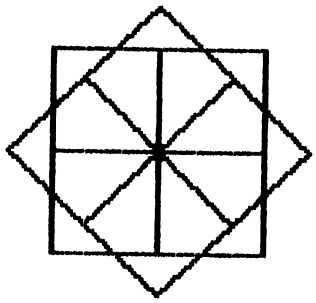
When the children have sufficiently dizzied themselves with circles and are ready for something new, suggest that they try to make a square. Return once again to the footprint activity. Select four children of approximately the same size to form a square with extended arms. Once again have one child walk around the figure while another places a footprint after each footstep. Ask the children to look at the square and to tell you what they see. They will probably mention that the sides have the same number of footprints and that there are four corners. Let the children go to the computer again to make a square. From the previous activities, they will probably know that a corner is 90 degrees. If not, let them discover it through trial and error. Have them make lots of squares of all different sizes and then combine them in whatever fashion they choose.

```
TO SQUARE
REPEAT 4 [FD 20 RT 90]
END

TO SQUARE2
REPEAT 4 [FD 40 RT 90]
END

TO SQUARE3
REPEAT 4 [FD 60 RT 90]
END
```

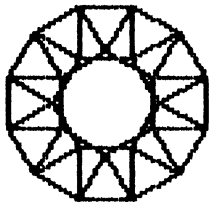








Ask the children if they can find the squares in this design.

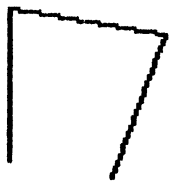


The triangle is a more difficult shape for children to describe. Help them solve the problem of finding the correct angle for an equilateral triangle by returning them to the V single-keystroke shape used in the previous pages. Change the name to B and then add one more command to the procedure so that it reads as follows.

```
TO B
FD 60
RT 150
FD 60
RT 150
FD 60
END
```

The procedure now has the three equal sides needed to make an equilateral triangle. However, the turning angle is too big causing two of the sides to cross over each other. Have the children try an angle of 100 to see what happens.

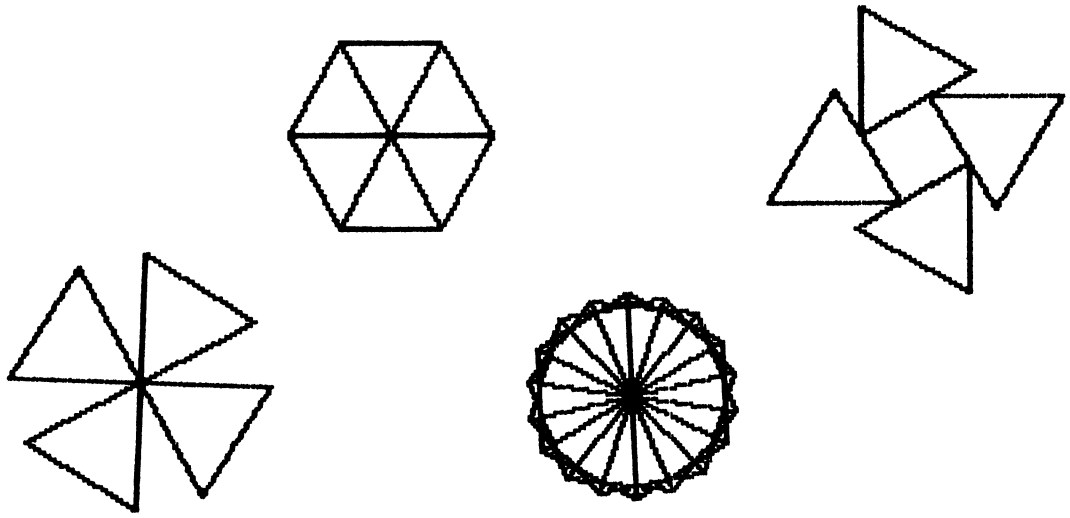
```
TO B2
FD 60
RT 100
FD 60
RT 100
FD 60
END
```



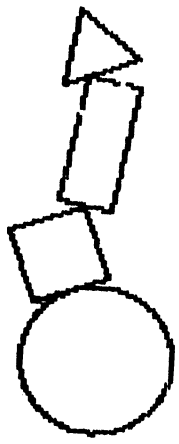
This time the angle is too small and the two sides don't meet. Allow the children to experiment with different angles until they discover the angle that will make an equilateral triangle. Once they find that RT 120 works perfectly, they can write a procedure using the REPEAT.

```
TO TRIANGLE
REPEAT 3 [FD 60 RT 120]
END
```

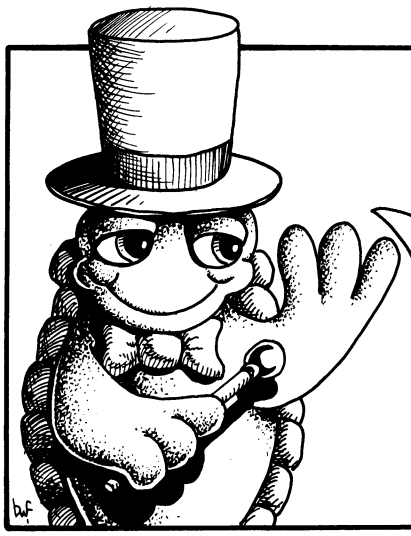
Just as the children explored different size circles and squares, encourage them to make several size triangles. Triangular designs are particularly interesting so they may want to spend quite a bit of time with these.



Finally, suggest to the children that they combine their circles, squares, and triangles into a single project. In the course of these projects the children may want to learn a new shape such as a rectangle, or they may want to include some of the single keystroke shapes learned earlier. Allow them as much freedom as possible within the bounds of what they can understand. On occasion a child may want to pursue a project that is too advanced. It is important to redirect his enthusiasm to a project that is manageable for him and one that he understands. A child who creates his own work will be proud of even simple projects; a child who merely types in commands is seldom satisfied with the results on the screen, however spectacular they might be.




YOU'RE PROBABLY ALL WONDERING WHY I CALLED YOU HERE TODAY...
©brad w foster 1984



Part III:
A First Look at List Processing

There is more to Logo than Turtle graphics. It has powerful list processing capabilities. Just as primary children can use Turtle graphics, so can they begin to use words and lists. The activities in this chapter require reading and spelling skills. They are also much more abstract than the graphics we have done so far. You can't see what's happening, even when you run a procedure. Thus it is extremely important to take your time. The commands will be introduced one at a time with activities to help the children understand them, use them and have fun with them.

The first command is PRINT. PRINT simply tells the computer to print something on the screen. It looks like this:

PRINT [TODAY IS MONDAY.]

The computer will print on the screen whatever you enclose in brackets. PRINT can be abbreviated PR. Many of the list processing commands can be abbreviated. Have the children use them in their complete form several times before introducing the abbreviation. Using it and repeating it in its full form will help them learn its meaning.

Have the children try some PRINT commands. Then put PRINT in a procedure and have some fun with it. After all, there's nothing really exciting about the computer simply retyping what we've just typed on the screen. Let's brag a little! Have each child make up a sentence about themselves like: Stephanie Rogers is wonderful! or Mike Nations is handsome! or Ashley Roberts is brilliant. Next, have them put the sentence in a procedure.

```
TO BRAG
PRINT [STEPHANIE ROGERS IS WONDERFUL!]
END
```

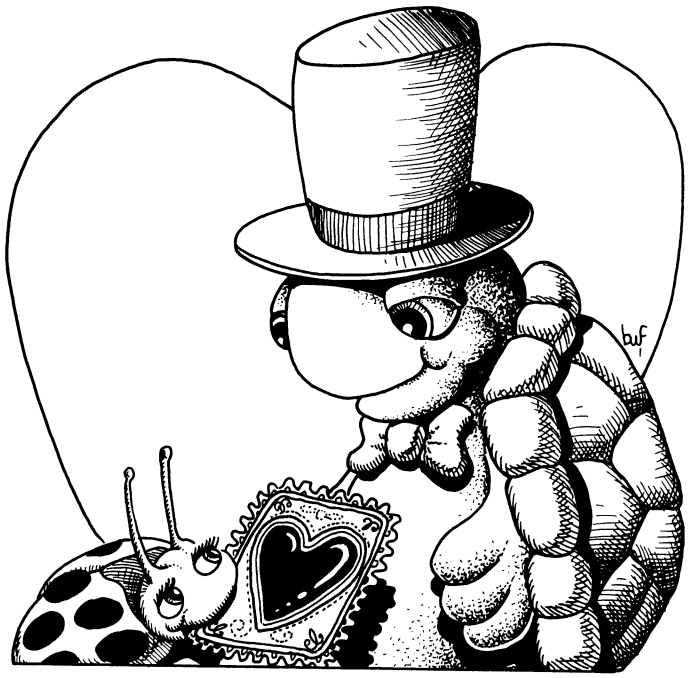
Try the procedure by typing BRAG and pressing RETURN. That's more fun, but let's make it into a BIG.BRAG. (Notice that you can use two words for the name of a procedure if you connect them with a period.)

```
TO BIG.BRAG
REPEAT 20 [BRAG]
END
```

Now when the command BIG.BRAG is given, the whole screen will be covered with STEPHANIE ROGERS IS WONDERFUL!

How about an instant valentine?

```
TO LOVE
PRINT [I LOVE MOM!]
END

REPEAT 20 [LOVE]
```



PRINT commands can be added to graphics procedures to identify them. For example, adding PRINT [CHRIS BAXTER'S BOAT] as the first command in a graphics procedure results in a picture with a name. Have the children add dates to their graphics procedures and they will begin to collect a chronological record of their work.

```
TO BOAT
HULL
SAIL
PRINT [CHRIS BAXTER'S BOAT]
PRINT [SEPTEMBER 20, 1984]
END
```

SENTENCE is a command that tells the computer to combine two lists into one sentence. SENTENCE can be abbreviated SE.

PRINT can be used with SENTENCE to have the computer combine two lists into one sentence. Here's an example of a PRINT SENTENCE command:

```
PRINT SENTENCE [LOGO IS] [A LANGUAGE FOR LEARNING.]
```

PRINT and SENTENCE can be used to combine more than two lists of words. However, if you want to combine three or more lists, it is necessary to put parentheses around SENTENCE and the lists:

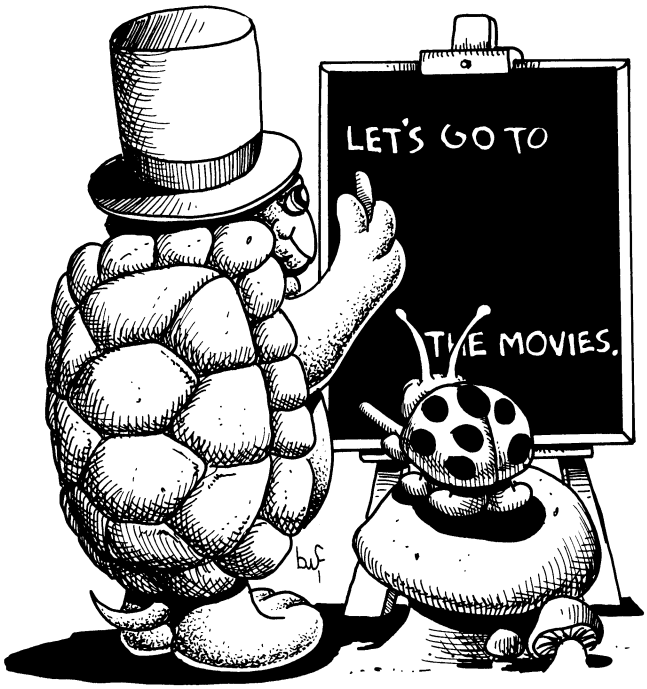
```
PRINT (SENTENCE [THE RAIN] [IN SPAIN] [FALLS MAINLY] [ON THE PLAIN.])
```

You may be wondering why the phrases in the above examples couldn't be combined already and printed with a simple PRINT command. They can. However, this is a way to have children practice with the SENTENCE command before it is combined with another command. The important concept to get across is that SENTENCE is used to combine two or more inputs. (You might liken it to sentence which includes a subject and a predicate, a noun and a verb.)

Just as with the PRINT command introduced earlier, it's not too exciting to see the computer retype something on the screen. It removes the brackets, but that's not much to hold anyone's attention. Let's use PRINT and SENTENCE in a sentence completion game.

Two children will take turns starting a sentence that the other has to complete. For example, the first child may type PRINT SENTENCE [I LIKE] and the second would have to complete the sentence, enclosed in brackets and with the correct punctuation mark. Put it in a procedure and it becomes even more fun because the sentences will be developed and then printed out as a group. Here's an example:

```
TO TALK
PRINT SENTENCE [I LIKE] [YOU.]
PRINT SENTENCE [I HATE] [TURNIPS.]
PRINT SENTENCE [LET'S GO TO] [THE MOVIES.]
PRINT SENTENCE [FLOWERS ARE] [PRETTY.]
PRINT SENTENCE [TURTLES ARE] [SLOW.]
PRINT SENTENCE [SCHOOL IS] [FUN.]
PRINT SENTENCE [YOU ARE] [FUNNY.]
END
```

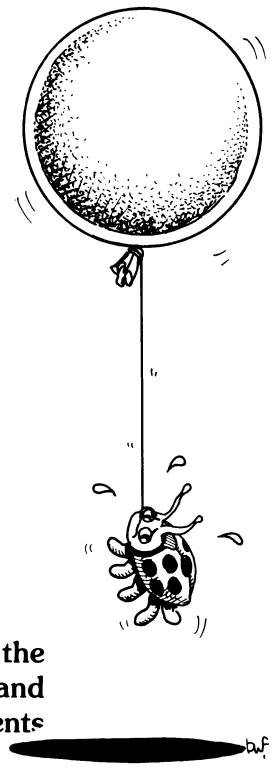


This activity could be repeated to develop paragraphs for a story. Before trying it on the computer, try an add-on story. The first child starts the story with a single sentence or part of a sentence. The second adds another thought, then the third, and so on until the entire class has contributed to the story.

Interactive Procedures

READLIST *(REQUEST for MIT)* is a command that makes the computer wait for someone to type an input. By using PRINT and SENTENCE with READLIST or *REQUEST.* you can develop interactive procedures which combine an input with a preprogrammed response.

```
TO CHAT
PRINT [WHAT IS YOUR NAME?]
PRINT SENTENCE [HELLO,] READLIST
PRINT [WHAT IS YOUR FAVORITE SPORT?]
PRINT SENTENCE [I DON'T LIKE] READLIST
PRINT [WHO'S YOUR BEST FRIEND?]
PRINT SENTENCE READLIST [IS SILLY!]
END
```



```
TO CHAT
PRINT [WHAT IS YOUR NAME?]
PRINT SENTENCE [HELLO,] REQUEST
PRINT [WHAT IS YOUR FAVORITE SPORT?]
PRINT SENTENCE [I DON'T LIKE] REQUEST
PRINT [WHO'S YOUR BEST FRIEND?]
PRINT SENTENCE REQUEST [IS SILLY!]
END
```

It seems like a conversation because the computer combines the answers with a pre-programmed response. Notice how READLIST and *REQUEST* can be used before or after the response. Have your students develop short conversations to try out on each other.

Sometimes, you might want the input to appear in the middle of a sentence. In that case, you will be combining three lists. (READLIST, and *REQUEST* are lists just like the lists enclosed in brackets.) Remember to use parentheses around SENTENCE and the lists when you combine three or more lists. For example:

```
PRINT [WHAT IS YOUR LUCKY NUMBER?]
PRINT (SENTENCE [THEN YOU CAN HAVE] READLIST [BALLOONS.])
```

```
PRINT [WHAT IS YOUR LUCKY NUMBER?]
PRINT SENTENCE [YOU JUST WON] REQUEST [BALLOONS.]
```

Have your students develop some questions and responses that will require combining three lists.

Suppose we want to ask several questions and then use the responses in a story or nonsense rhyme. We want to make the computer remember the responses and use them later, not right away. We'll have to MAKE the response into something — give it a label or a name — so the computer can store it and call it back up. To do that, we use the MAKE command. Here is an example. Suppose we ask someone "What is your favorite color?"

```
PRINT [WHAT IS YOUR FAVORITE COLOR?]
```

We'll have the computer store the answer under the label COLOR, like this:

```
MAKE "COLOR READLIST
```

```
MAKE "COLOR REQUEST
```

Later we'll use the answer as part of a story, like this:

```
PRINT SENTENCE [THE KNIGHT'S BANNER WAS]  :COLOR
```

Notice that when you give the answer a label, you use single quotation marks. When you want to use the answer as part of a sentence, you use a colon.

Parts of Speech

This is an activity that students really enjoy while they're learning to construct sentences. Start out with a shoe box divided into four sections. The sections should be labeled with Adjective, Noun, Verb and Adverb.

Give slips of paper to four students. The first will write down an adjective and put it in the section labeled adjective. The second will write down a noun and put it in the noun section. The third will write down a verb and put it in the verb section, and the fourth will write down an adverb and put it in the adverb section. A fifth student will play the role of the computer and read the slips of paper one at a time, composing a silly sentence.

Think about the steps and figure out how to do the same thing on the computer. First define a procedure to ask someone to think of an adjective, a noun, a verb and an adverb and store them under labels so the computer can remember them and call them back up. They have the computer combine them into a sentence. (CLEARTEXT is used to clear the screen before the final "creation" is printed.)

```
TO SILLY.SENTENCE
PRINT [NAME AN ADJECTIVE]
MAKE "ADJECTIVE READLINE
PRINT [NAME A NOUN]
MAKE "NOUN READLINE
PRINT [NAME A VERB]
MAKE "VERB READLINE
PRINT [NAME AN ADVERB]
MAKE "ADVERB READLINE
CLEARTEXT
PRINT (SENTENCE  :ADJECTIVE  :NOUN  :VERB  :ADVERB)
END
```

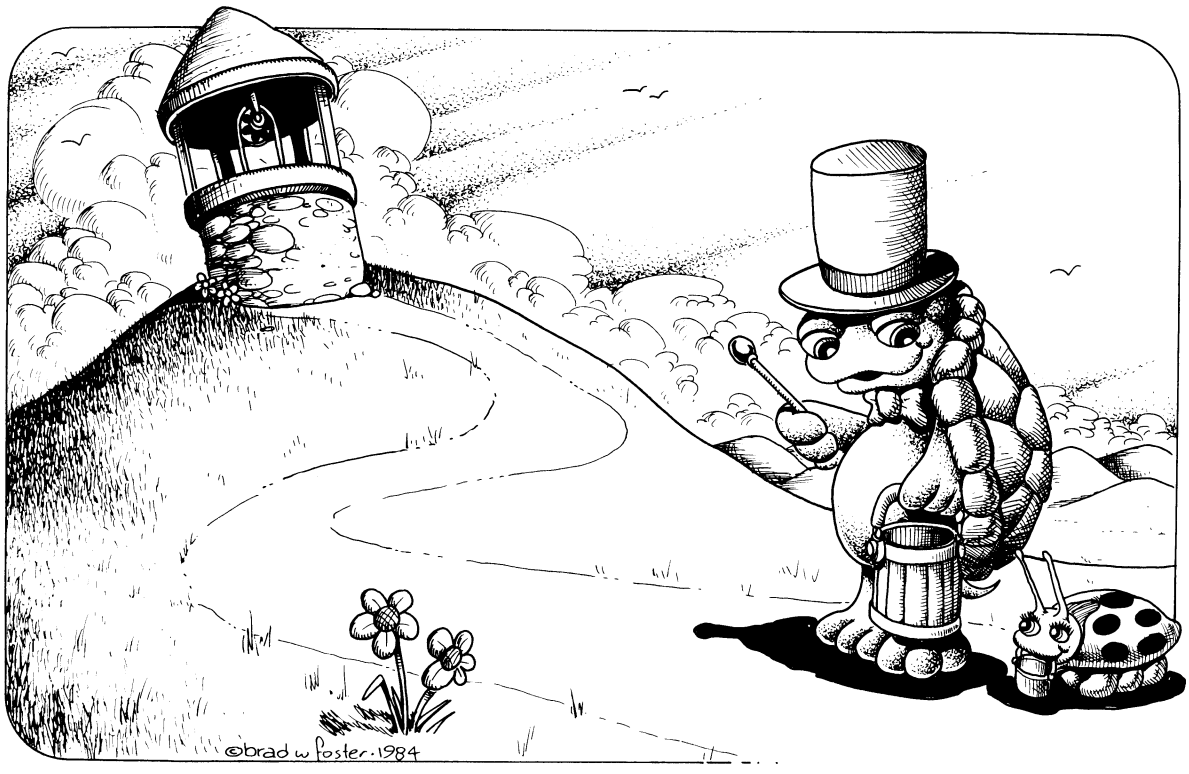
```
TO SILLY.SENTENCE
PRINT [NAME AN ADJECTIVE]
MAKE "ADJECTIVE REQUEST
PRINT [NAME A NOUN]
MAKE "NOUN REQUEST
PRINT [NAME A VERB]
MAKE "VERB REQUEST
PRINT [NAME AN ADVERB]
MAKE "ADVERB REQUEST
CLEARTEXT
PRINT (SENTENCE  :ADJECTIVE  :NOUN  :VERB  :ADVERB)
END
```

Once the students have played with the procedure, they will probably come up with ways to improve it. They may wish to ask for plural nouns, for example.

Madlibs

Let's use PRINT, SENTENCE, READLIST or *REQUEST* and MAKE to bring new life to some old nursery rhymes and stores. Have the children pick out the key words in a familiar nursery rhyme. Then have them make up questions which could be answered with those words. For example, two questions for "Jack and Jill" might be "Name a boy." and "Name a girl." Put the questions into a procedure and have the computer combine responses with the rest of the nursery rhyme. Here's what "Jack and Jill" might look like: (CLEARTEXT is used to clear the screen before the final "creation" is printed.)

```
TO STORY
PRINT [NAME A BOY.]
MAKE "BOY READLIST  (REQUEST)
PRINT [NAME A GIRL.]
MAKE "GIRL READLIST  (REQUEST)
PRINT [NAME SOMETHING TO CLIMB ON.]
MAKE "CLIMB READLIST  REQUEST)
PRINT [NAME A CONTAINER.]
MAKE "CONTAINER READLIST  (REQUEST)
PRINT [NAME SOMETHING TO DRINK.]
MAKE "DRINK READLIST  (REQUEST)
PRINT [NAME A PART OF THE BODY.]
MAKE "BODY READLIST  (REQUEST)
PRINT [NAME A VERB ENDING WITH ING.]
MAKE "VERB READLIST  (REQUEST)
CLEARTEXT
PRINT (SENTENCE  :BOY [AND]  :GIRL)
PRINT SENTENCE [WENT UP THE]  :CLIMB
PRINT (SENTENCE [TO FETCH A]  :CONTAINER [OF]  :DRINK)
PRINT SENTENCE  :BOY [FELL DOWN.]
PRINT SENTENCE [AND BROKE HIS]  :BODY
PRINT (SENTENCE [AND]  :GIRL [CAME]  :VERB [AFTER.] )
END
```



Quizzes

IFTRUE and IFFALSE are conditional commands. A conditional statement includes two parts, the IF part and the THEN part. Have the children think of examples of conditional statements. If you clean your room, you can go play. If it's raining, bring your umbrella. How about keeping a list of all the conditional statements they hear in one day?

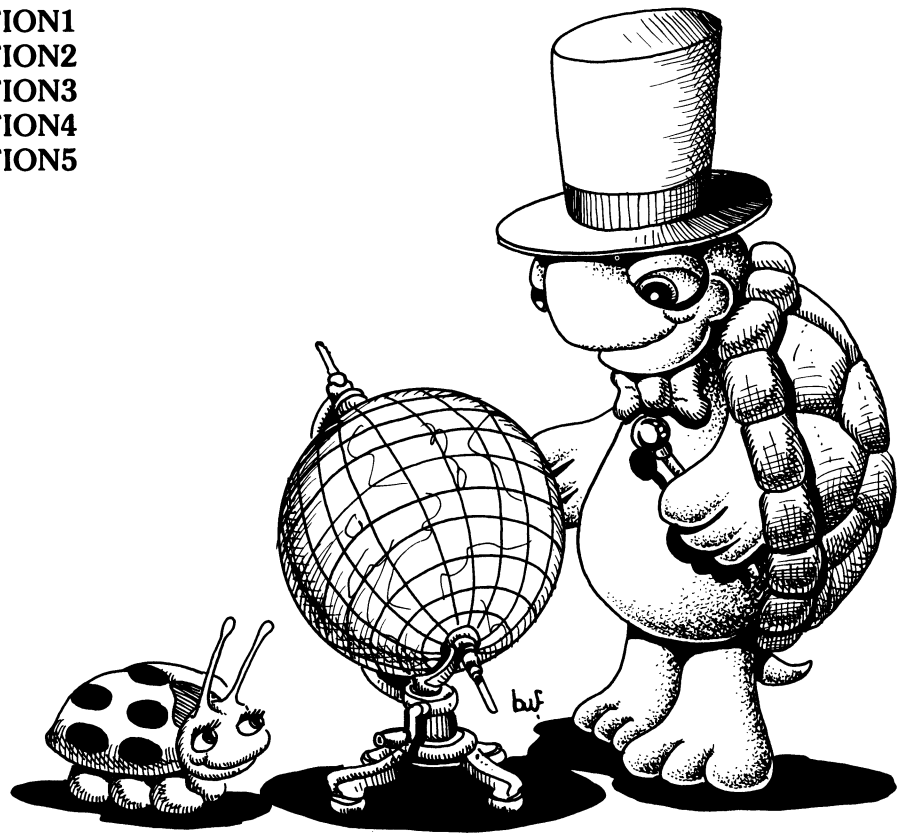
To use IFTRUE and IFFALSE commands, we'll set up a procedure that asks for a response. Then we will TEST the response to see if it is correct. IF it is correct, or TRUE, the computer will carry out one set of instructions. IF it is FALSE, the computer will carry out a different set of instructions.

```
TO QUESTION
PRINT [WHAT IS THE CAPITAL OF TEXAS?]
TEST READLIST = [AUSTIN]
IFTRUE [PRINT [THAT'S CORRECT!]]
IFFALSE [PRINT [NO, THE CAPITAL OF TEXAS IS AUSTIN.]]
END
```

TO QUESTION
PRINT [WHAT IS THE CAPITAL OF TEXAS?]
TEST REQUEST = [AUSTIN]
IFTRUE PRINT [THAT'S CORRECT!]
IFFALSE PRINT [NO, THE CAPITAL OF TEXAS IS AUSTIN.]
END

Pick a subject you are currently studying and have the children make up questions about it. Then put them into a QUIZ. Two or three children could work on individual procedures and then they could be combined into a super procedure.

```
TO QUIZ
QUESTION1
QUESTION2
QUESTION3
QUESTION4
QUESTION5
END
```



The Interactive, Illustrated, All-Purpose Adventure Story

Writing a story and illustrating it is a project that offers a variety of experiences and explorations. We'd like to conclude this chapter with an example of one such story and how it grew from a simple idea into a project that included word processing, research, geometry, geography, and interactive procedures written in Logo. Our example was developed and written by Stacy Bearden, age 9. The only preliminary instructions were to write a story about Toulouse and Lady B, the characters we have used throughout this book.

With the few commands we have learned so far, we could write simple stories and illustrate them with Turtle graphics. Rather than start with Logo, however, now might be a good time to introduce word processing to your students. This is a way to include a variety of computer uses in a single project. Word processing programs are relatively inexpensive and make it easy to write stories on the computer. Sentences and paragraphs can be moved quite easily and misspelled words corrected without having to retype the entire manuscript.

GOING TO CALIFORNIA
by Stacy Bearden

Well, here we are in California. You might as well know how we got here. As I say, when approaching a big leaf, better start at the top.

My name is Lady B. and this is my friend Toulouse. I'm a ladybug and Toulouse is a turtle. One day, we decided to visit our friend, Danielle, in California.

We started out in Texas and pretty soon we were in New Mexico. "I hear there are beautiful caves somewhere in New Mexico," Toulouse said. We walked on. Soon we met a butterfly sitting on a flower.

"Excuse me," I said, "do you happen to know where the beautiful caves people talk about are?" "It so happens I do, follow me."

So we followed the butterfly (whose name was Daffodil) all day long.

Finally we came to some gigantic caves. "Wow, can we go in them?" Toulouse wanted to know. "Sure," Daffodil said, "but if you're in a hurry..." "No, we'll go in!" I interrupted. "I'll have to leave you now because I can't bear the darkness." "Bye," we shouted as we ran to find a tour guide.

When we got out, we decided it was time to move on. We headed west and soon came to Arizona. Still we traveled on until we came upon a steep ledge. We had come to the Grand Canyon. What a pretty sight! At the bottom, a river was running through it. The sunburned rocks glowed red when the sunbeams reached them.

"We have to be moving on," I said sadly. "We'll be coming to California soon." "Can we go to Disneyland?" Toulouse asked eagerly. "Of course we can," I answered. "After all, we didn't come here for nothing."

When we got to California, Danielle was there waiting. "What took you so long," she asked. "We went to Carlsbad Caverns and the Grand Canyon on the way," Toulouse answered.

In order to write the story, Stacy had to figure out which states to go through. She could have looked it up in an Atlas, but she chose to work a jigsaw puzzle of the United States. On the back of the U.S. puzzle was the world, so the simple trip to California also included locating entire continents.

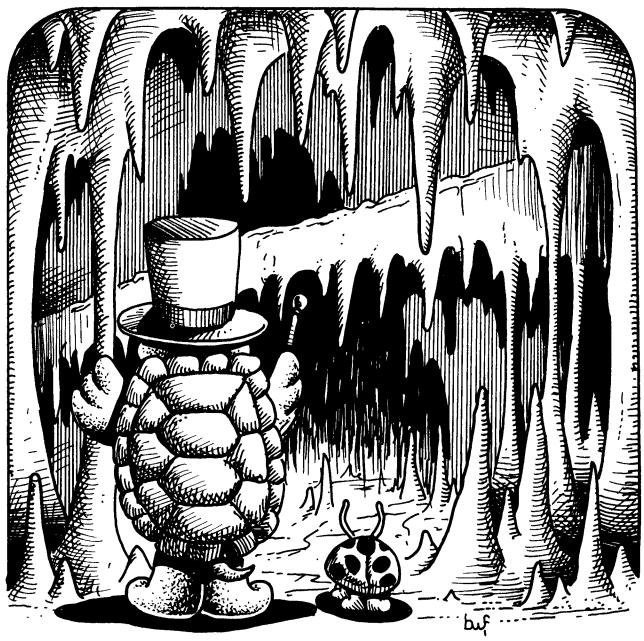
After writing an adventure story as a story, let's look at some ways we could break it down into Logo procedures, both text and graphics. Perhaps we could start by asking the reader some geography questions.

```
TO GEOGRAPHY
PRINT [IF WE WERE GOING FROM TEXAS TO CALIFORNIA,]
PRINT [WHAT IS THE FIRST STATE WE WOULD COME TO?]
TEST READLIST= [NEW MEXICO]
IFTRUE [PRINT [RIGHT!]]
IFFALSE [PRINT [GO LOOK AT A MAP!] GEOGRAPHY]
PRINT [WHAT FAMOUS CAVE IS IN NEW MEXICO?]
TEST READLIST = [CARLSBAD CAVERNS]
IFTRUE [PRINT [YES!]]
IFFALSE [PRINT [NO, IT'S CARLSBAD CAVERNS.]
END
```

```
TO GEOGRAPHY
PRINT [IF WE WERE GOING FROM TEXAS TO CALIFORNIA,]
PRINT [WHAT IS THE FIRST STATE WE WOULD COME TO?]
TEST REQUEST = [NEW MEXICO]
IFTRUE PRINT [RIGHT]
IFFALSE PRINT [GO LOOK AT A MAP!] GEOGRAPHY
PRINT [WHAT FAMOUS CAVE IS IN NEW MEXICO?]
TEST REQUEST = [CARLSBAD CAVERNS]
IFTRUE PRINT [YES!]
IFFALSE PRINT [NO, IT'S CARLSBAD CAVERNS.]
```

We could add some information or even drawings of Carlsbad Caverns. Let's add a procedure about crystal formations.

```
TO QUIZ
QUESTION1
QUESTION2
END
```



```
TO QUESTION1
CLEARTEXT TEXTSCREEN
PRINT [WHAT IS A STALACTITE?]
PRINT [TYPE A, B, OR C.]
PRINT [ ]
PRINT [A. A FORMATION THAT HANGS FROM THE ROOF OF A CAVE.]
PRINT [B. A PAINFUL CAVITY.]
PRINT [C. A MALE DEER WITH A FULL HEAD OF ANTLERS.]
TEST READLIST = [A]
IFTRUE [PRINT [RIGHT!] WAIT 50]
IFFALSE [PRINT [NO, LET ME SHOW YOU.] WAIT 50]
STALACTITE
END
```

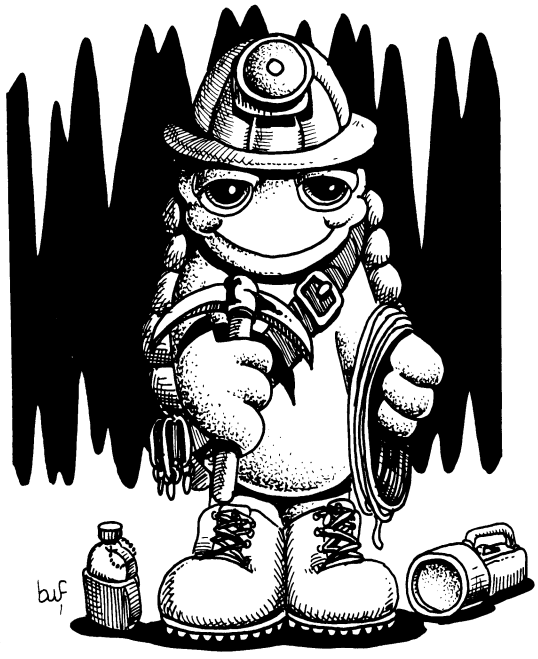
```
TO QUESTION1
CLEARTEXT TEXTSCREEN
PRINT [WHAT IS A STALACTITE?]
PRINT [TYPE A, B, OR C.]
PRINT [ ]
PRINT [A. A FORMATION THAT HANGS FROM THE ROOF OF A CAVE.]
PRINT [B. A PAINFUL CAVITY.]
PRINT [C. A MALE DEER WITH A FULL HEAD OF ANTLERS.]
TEST REQUEST = [A]
IFTRUE PRINT [RIGHT!] WAIT 50
IFFALSE PRINT [NO, LET ME SHOW YOU.] WAIT 50
STALACTITE
END
```

For MIT Logo, you will have to define WAIT.

```
TO WAIT :TIME
IF :TIME = 0 STOP
WAIT :TIME - 1
END
```

Define a procedure to draw a cave and add some crystals.

```
TO STALACTITE
CS HT
PRINT [A STALACTITE IS A CRYSTAL FORMATION THAT]
PRINT [HANGS FROM THE ROOF OF A CAVE.]
CAVE
PRINT [A STALAGMITE IS A CRYSTAL FORMATION THAT]
PRINT [GROWS UP FROM THE FLOOR OF THE CAVE.]
CAVE2
WAIT 100 CS
END

TO CAVE
PU SETX - 70 PD
REPEAT 19 [FD 10 RT 10]
RT 80 FD 115
PU SETX - 70 SETY 0 SETH 0 PD
REPEAT 6 [FD 10 RT 10]
REPEAT 8 [CRYSTAL FD 10 RT 10]
REPEAT 5 [FD 10 RT 10]
WAIT 50
END

TO CRYSTAL
RT 90 FD 15 BK 15 LT 90
END

TO CAVE2
RT 80
REPEAT 8 [FD 12 TRI]
END

TO TRI
REPEAT 3 [FD 7 RT 120]
END
```



```
TO QUESTION2
CLEARTEXT TEXTSCREEN
PRINT [WHAT IS A PERSON CALLED WHO STUDIES CAVES?]
PRINT [TYPE A, B, OR C.]
PRINT [ ]
PRINT [A. A CAVEMAN]
PRINT [B. A SPELUNKER]
PRINT [C. AN AVIATOR]
TEST READLIST = [B]
IFTRUE [PRINT [VERY GOOD!]]
IFFALSE [PRINT [TRY AGAIN!] WAIT 100 QUESTION2]
END
```

```
TO QUESTION2
CLEARTEXT TEXTSCREEN
PRINT [WHAT IS A PERSON CALLED WHO STUDIES CAVES?]
PRIUNT [TYPE A, B, or C.]
PRINT [ ]
PRINT [A. A CAVEMAN]
PRINT [B. A SPELUNKER]
PRINT [C. AN AVIATOR]
TEST REQUEST = [B]
IFTRUE PRINT [VERY GOOD!]
IFFALSE PRINT [TRY AGAIN!] WAIT 100 QUESTION2
END
```

When QUESTION1 is answered, either correctly or incorrectly, the computer goes on to a visual representation of a cave with stalactites and stalagmites. Then the screen is cleared and QUESTION2 appears. QUESTION2 will be repeated until the student chooses the correct answer. Another graphic procedure could be included here or we could go on to more questions and answers.

Similar procedures could be written to give out information or ask questions about the Grand Canyon. Where is the Grand Canyon? What river flows through it? Maybe someone would like to try to draw a donkey to take people down the trail.

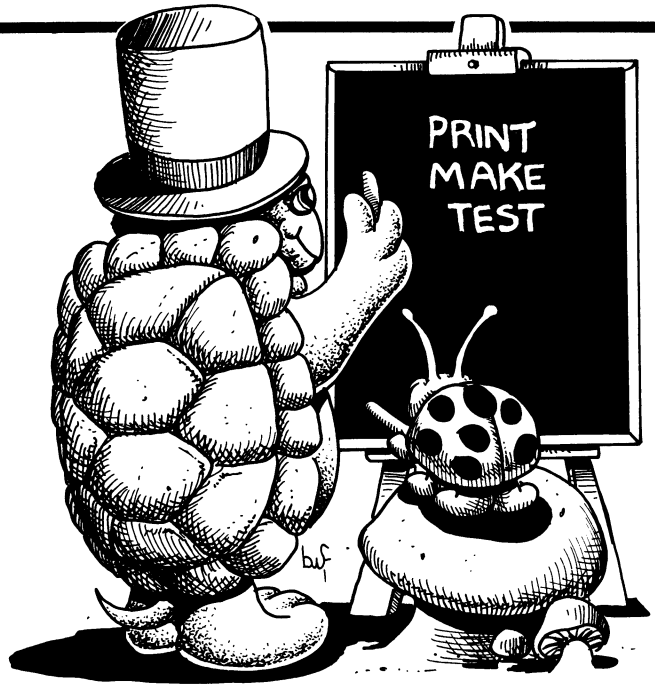
The characters in the story ended up at Disneyland. What a great place to end up! You could challenge your students to have the Turtle design a new ride. Or how about a fireworks display complete with changing colors? Or a mouse with big ears?

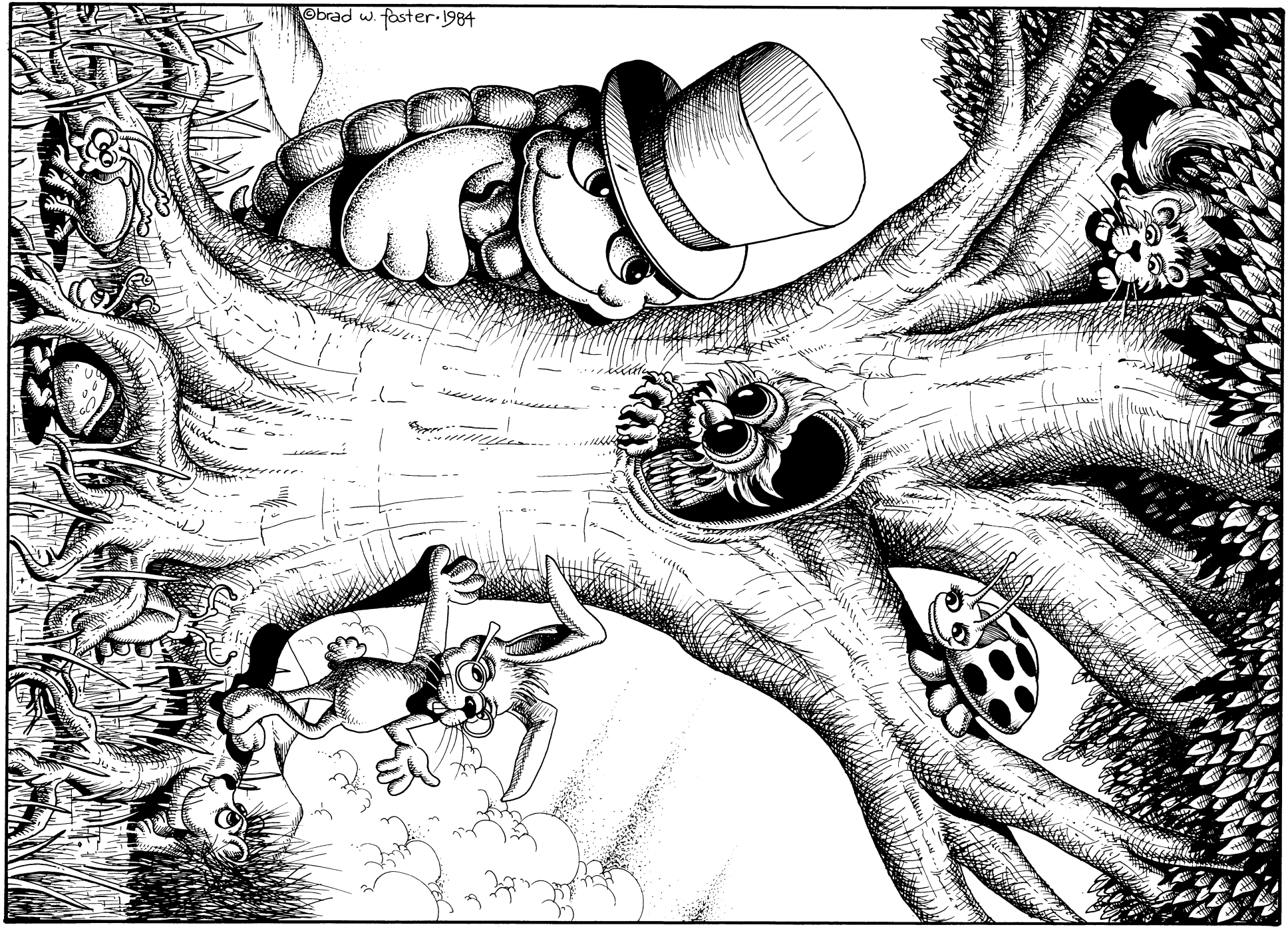
This is a project that could go on and on since it has the possibility of including so many different subjects: language arts, spelling, geography, research, mathematics, creative writing, science, and so forth. Each new discovery can lead to a whole new area of exploration. The only limitation will be the amount of memory of your particular computer. If you run out of space in the memory, break the project into parts, storing each part as a separate file on your disk.

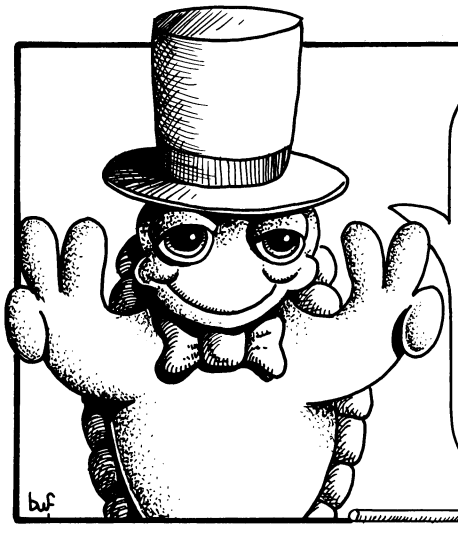
List Processing Commands

(as used in Primarily Logo)

PRINT [ ] —	Whatever is in the brackets will be printed on the screen.
SENTENCE —	Tells the computer to combine two or more lists of words.
READLIST (LCSI) — REQUEST (MIT)	Waits for someone to type something. Uses what is typed in a sentence.
MAKE —	Makes the computer remember an input so it can be used later.
TEST —	Tells the computer to check an answer.
IFTRUE —	Tells the computer to do something if the answer matches a pre-programmed response.
IFFALSE —	Tells the computer to do something else if the answer doesn't match the pre-programmed response.




©brad w. foster·1984

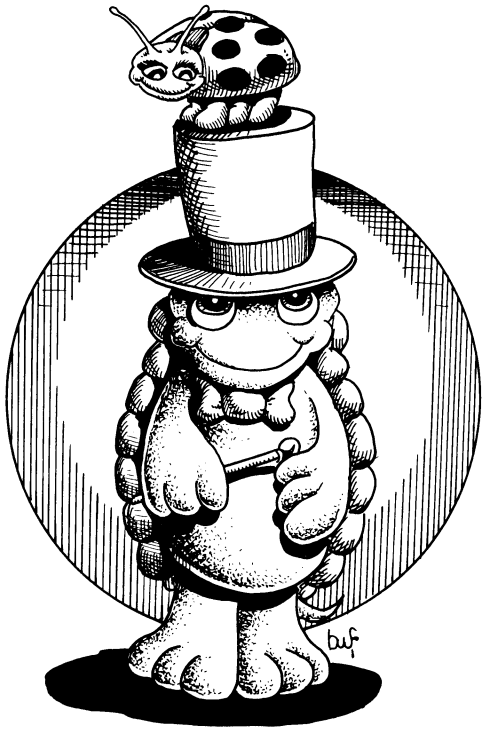


Part IV: Instructional Tasks

We will now turn our attention to programs to help children develop further understanding of simple spatial relationships, colors, and fundamental arithmetic operations. As models of interactive programs, they are designed to give the child control of what happens on the screen within the limits of a specific vocabulary. These are not programs written by children, but programs written for children. The programs can be expanded, modified, or truncated to serve individual needs.

Spatial Relations

While still quite young, children begin to describe their spatial relationship to things and the spatial relation of one thing to another. One block is **on top of** another; a toy is **under** the bed; a shoe is **next to** the sofa. Hands are **inside** of pockets or **behind** the back. Children also describe the motions that they feel in their own bodies and those that they see in the bodies around them. They jump **up** and **down** and run **around.** The ball bounces **over** the fence; the car is driven **into** the garage; the train goes **through** the tunnel.



The Logo procedure entitled GETGOING allows children to direct the Turtle around the screen using commands that represent different spatial relations. It's a good idea to prepare the children for the program through some off-computer activities.

The first activity is patterned on the popular game of "Simon Says." Cut triangular Turtles out of stiff paper or cardboard. (See pattern in Appendix A.) Using a thumbtack, attach each Turtle to the eraser end of a pencil and give one to every child. Ask the children to make a circle with one child in the center. The center child calls out the commands using the format: The computer says, "Move the Turtle _____." The commands inserted in the blank should correspond to those in the program.

Commands included in GETGOING

NORTH	SOUTH	EAST	WEST	AROUND
FASTER	SLOWER		STOP	GO

Remember that you can modify the program to include any number of commands. The children should follow commands preceded by "The computer says..." and ignore the others. Those who follow the correct commands remain in the game, while the others do not.

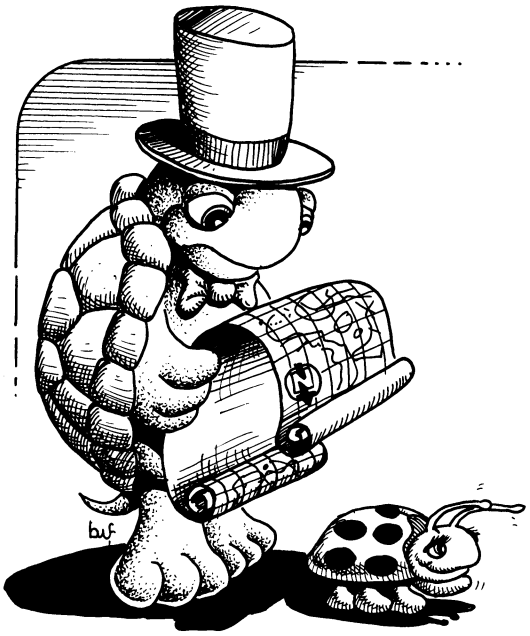
The "Computer Says" game enables a large group of children to participate in playing Turtle. If you are working with children individually or in a learning center situation, a similar experience can be provided using a large mirror and a much smaller triangular Turtle attached to the pencil eraser. Label the four sides of the mirror with NORTH, SOUTH, EAST, and WEST. Prop the mirror in front of the child at an angle that allows him to see his chest but not his face. Have him hold the pencil about 4 inches from the mirror. The Turtle will be reflected in the mirror and its motions can be easily seen.

Seat a second child so that he can see the mirror while giving commands to the child with the Turtle. If the command is "north," the first child begins moving the Turtle toward the top of the mirror until the second child gives another command. He can continue to give commands at will as long as the movements of the Turtle can still be seen in the mirror. The first child must be sure to turn the Turtle in the correct direction in response to each command.

A variation of the activity involves attaching objects to the mirror. For example, a cut-out of a square could be taped onto the mirror. The commands could then include "on top of," "next to," "under," and "into." The addition of a second square cut-out could allow the Turtle to go "between" or "through" the opening.

Another activity that is simple but effective with very young children is for one of the children to "play Turtle." The child gets down on all fours and can be given some kind of Turtle attire. A box with holes cut for hands, head, and feet works well. The other children take turns giving the Turtle commands. The Turtle can be told to crawl "under" a table or "over" a stack of books, to go "around" in a circle or "next to" a desk, to move "between" two students or "through" a door.

Once the children have had substantial experience with spatial relation activities off the computer, they are ready to attempt a more abstract level of activity on the computer. The GETGOING program is a popular model that can be easily adapted. The "guts" of the program is the procedure called COMMAND.



```
TO COMMAND
MAKE "COM READKEY
IF :COM = "NORTH [SETH 0]
IF :COM = "SOUTH [SETH 180]
IF :COM = "EAST [SETH 90]
IF :COM = "WEST [SETH 270]
END
```

TO COMMAND
MAKE "COM READKEY
IF :COM = "NORTH SETH 0
IF :COM = "SOUTH SETH 180
IF :COM = "EAST SETH 90
IF :COM = "WEST SETH 270
END

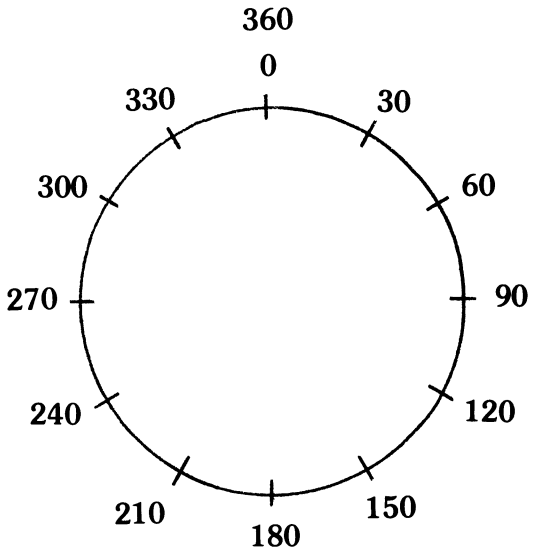
READKEY is called a tool procedure. These are simply procedures that help out other procedures. Alone READKEY is of no value; it works only in conjunction with another procedure.

```
TO READKEY
IF KEYP [CLEARTEXT OUTPUT READWORD]
OUTPUT "
END
```

TO READKEY
IF RC? CLEARTEXT OUTPUT FIRST REQUEST
OUTPUT "
END

READKEY waits for a command to be typed, clears any text off the screen, prints the command, and then allows the selected command from within COMMAND to be executed.

These four initial commands describe the Turtle's relationship to a maplike screen. If the command is NORTH (IF :COM = "NORTH), the Turtle sets its heading at 0 (SETH 0). Headings are based on the circle, or 360 degrees.



The COMMAND procedure, which checks to see what has been typed and then carries out the command, is contained within the GETGOING procedure.

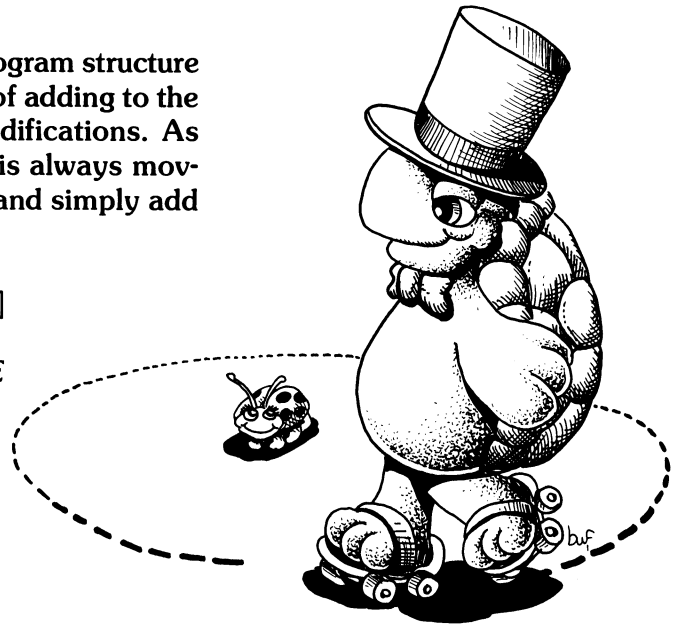
```
TO GETGOING :DISTANCE
FD :DISTANCE
COMMAND
GETGOING :DISTANCE
END
```

GETGOING is what is known as a **recursive** procedure, one which calls itself over and over until it is stopped with a STOP command or manually. Without it, COMMAND would allow you to type in just one command and then it would stop. GETGOING starts the Turtle moving, implements whatever is requested through the COMMAND procedure, and keeps the Turtle moving. The distance is a variable in the procedure. This variable allows the Turtle to move at different speeds. With GETGOING 2 the Turtle moves twice as fast as with GETGOING 1.

With these three procedures, the program structure is in place. Now it is simply a matter of adding to the program and/or making desired modifications. As GETGOING now stands, the Turtle is always moving. Let's keep it moving for a while and simply add more commands.

IF :COM = "AROUND [CIRCLE]

IF :COM = "AROUND CIRCLE



If the command AROUND is typed, the procedure CIRCLE should be followed. Before the procedure can be followed it must be written.

```
TO CIRCLE
REPEAT 120 [FD 1 RT 3]
END
```

Now if you type AROUND, the Turtle will make a circle. If you want the Turtle to make a larger circle, just modify the CIRCLE procedure.

What about changing the Turtle's speed? Remember that GETGOING has a variable distance which enables the Turtle to move FASTER or SLOWER.

```
IF :COM = "FASTER [MAKE "DISTANCE :DISTANCE + 1]
IF :COM = "SLOWER [CHECK]
```

```
IF  :COM = "FASTER MAKE "DISTANCE  :DISTANCE + 1
IF  :COM = "SLOWER CHECK
```

If the program is started with GETGOING 1, FASTER makes the distance change from 1 to 2. The Turtle would now be moving two steps at a time instead of only one. Type in FASTER again and the Turtle would move three steps at a time.

CHECK enables us to slow down the Turtle until it gets so slow it stops.

```
TO CHECK
TEST  :DISTANCE > 0
IFTRUE [MAKE "DISTANCE  :DISTANCE - 1]
IFFALSE [STOP]
END
```

```
TO CHECK
TEST  :DISTANCE > 0
IFTRUE MAKE "DISTANCE  :DISTANCE - 1
IFFALSE STOP
END
```

We can also stop the motion of the Turtle by adding a STOP command.

```
IF :COM = "STOP [MAKE "DISTANCE 0]
```

```
IF  :COM = "STOP MAKE "DISTANCE 0
```

Setting the distance at 0 will STOP the Turtle. Once the Turtle has stopped, however, you will need to add a GO command to get it going once again.

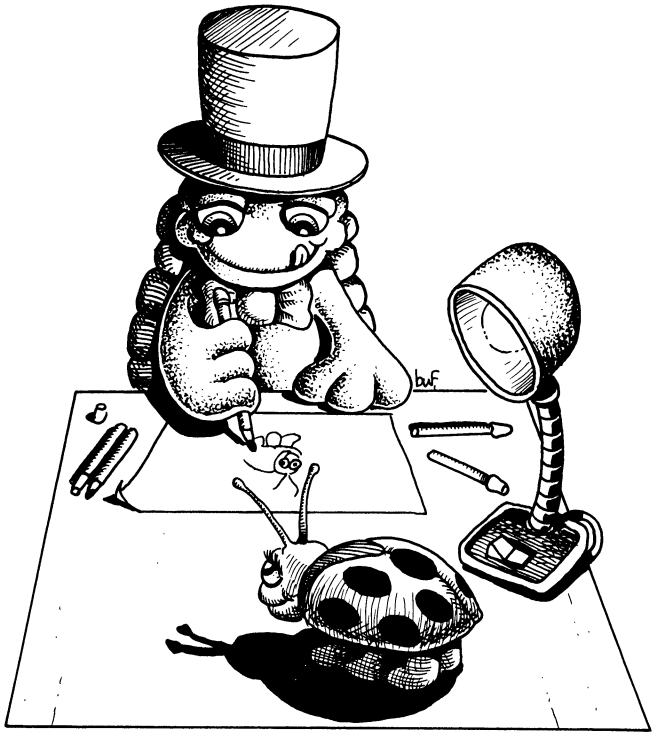
```
IF :COM = "GO [MAKE "DISTANCE 1]
```

```
IF  :COM = "GO MAKE "DISTANCE 1
```

The COMMAND procedure thus far reads as follows.

```
TO COMMAND
MAKE "COM READKEY
IF :COM = "NORTH [SETH 0]
IF :COM = "SOUTH [SETH 180]
IF :COM = "EAST [SETH 90]
IF :COM = "WEST [SETH 270]
IF :COM = "AROUND [CIRCLE]
IF :COM = "FASTER [MAKE "DISTANCE :DISTANCE + 1]
IF :COM = "SLOWER [CHECK]
IF :COM = "STOP [MAKE "DISTANCE 0]
IF :COM = "GO [MAKE "DISTANCE 1]
END
```

```
TO COMMAND
MAKE "COM READKEY
IF :COM = "NORTH SETH 0
IF :COM = "SOUTH SETH 180
IF :COM = "EAST SETH 90
IF :COM = "WEST SETH 270
IF :COM = "AROUND CIRCLE
IF :COM = "FASTER MAKE "DISTANCE :DISTANCE + 1
IF :COM = "SLOWER CHECK
IF :COM = "STOP MAKE "DISTANCE 0
IF :COM = "GO MAKE "DISTANCE 1
END
```



The pen is up (PU) in the procedure to prevent the Turtle from drawing or leaving a trail. The PENUP command can be removed (or replaced with PENDOWN if you wish the line of movement to be visible). However, it is important to remember that even so simple a change can result in the program serving a very different purpose. With the PENDOWN (PD) the focus shifts to what is drawn or the consequences of movement rather than the movement itself. It is important to always ask: How does a change in a program affect the purposes or intentions of the program? Programming itself can be such an exciting endeavor that sometimes we can be mesmerized by the effects on the screen and forget what effect it was that we really wanted.

Suppose you want to use the program to work with spatial relations such as "on top of," "next to," and "between." How would the program need to be changed? First, there will need to be some objects on the screen to be "on top of," "next to," or "between." These can be attached to the screen or drawn by the Turtle. Second, the Turtle may need to be easier to control than the present program allows.

Let's attend to the second problem first. If the Turtle would change direction upon command but wait for a second command before moving, it would be easier to position it accurately. Adding a "MAKE DISTANCE 0 after the appropriate commands will stop the Turtle and make it wait for the GO command. You might also want to eliminate the FASTER and SLOWER commands from the program at this point. (In the appendix, the modified procedure is called GETGOING.AGAIN.) GETGOING.AGAIN 1 should provide an appropriate speed.

```
TO COMMAND
MAKE "COM READKEY
PU
IF :COM = "NORTH [SETH 0 MAKE "DISTANCE 0]
IF :COM = "SOUTH [SETH 180 MAKE "DISTANCE 0]
IF :COM = "EAST [SETH 90 MAKE "DISTANCE 0]
IF :COM = "WEST [SETH 270 MAKE "DISTANCE 0]
IF :COM = "AROUND [CIRCLE MAKE "DISTANCE 0]
IF :COM = "STOP [MAKE "DISTANCE 0]
IF :COM = "GO [MAKE "DISTANCE 1]
END
```

```
TO COMMAND
MAKE "COM READKEY
PU
IF :COM = "NORTH SETH 0 MAKE "DISTANCE 0
IF :COM = "SOUTH SETH 180 MAKE "DISTANCE 0
IF :COM = "EAST SETH 90 MAKE "DISTANCE 0
IF :COM = "WEST SETH 270 MAKE "DISTANCE 0
IF :COM = "AROUND CIRCLE MAKE "DISTANCE 0
IF :COM = "STOP MAKE "DISTANCE 0
IF :COM = "GO MAKE "DISTANCE 1
END
```

Now let's tackle the problem of putting objects on the screen. Boxes are easy to make, so let's put a box on the screen and shade it in.

```
TO SQUARE
PD
REPEAT 4 [FD 20 RT 90]
REPEAT 10 [FD 20 RT 90 FD 1 RT 90 FD 20 LT 90 FD 1 LT 90]
PU
END
```

The first REPEAT draws the box, while the second REPEAT fills it in. If more than one box is needed a second can be added. The box-making command needs to be inserted within the COMMAND procedure.

```
IF :COM = "BOX [SQUARE MAKE "DISTANCE 0]
```

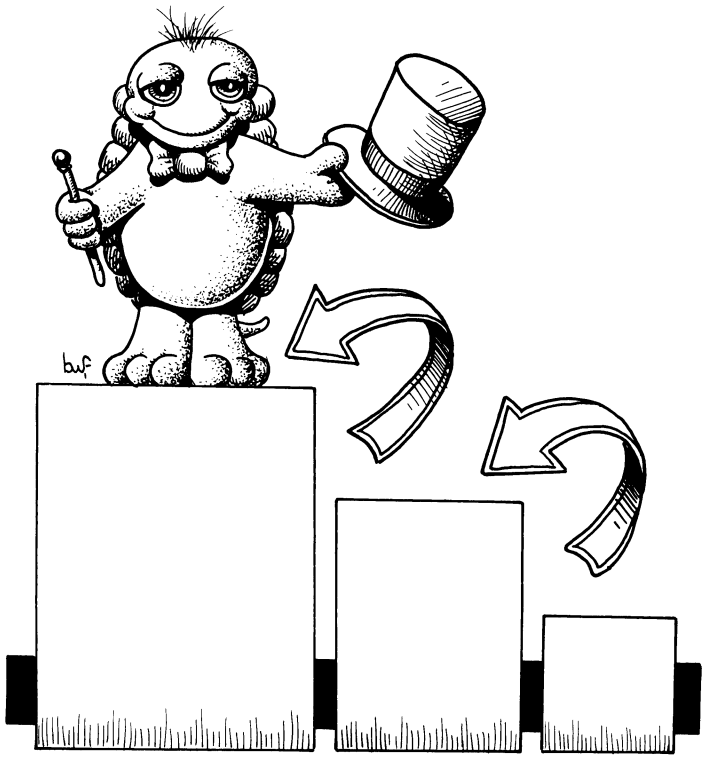
```
IF :COM = "BOX SQUARE MAKE "DISTANCE 0
```

If you want to clear the screen, a CLEAR command is needed.

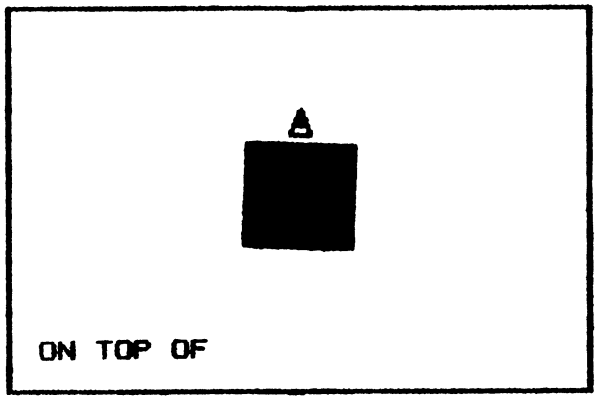
```
IF  :COM = "CLEAR [CLEARSCREEN]
```

IF :COM = "CLEAR CLEARSCREEN

As more and more commands are added to the program, it tends to become congested and begins to slow down. An easy way to solve the problem is to add a STOP at the end of each of the commands. (This is done in the GETGOING.AGAIN procedure in the appendix.)



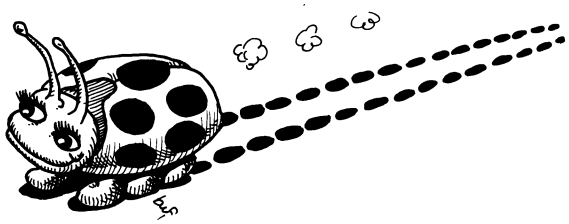
If the command ON TOP OF were added to the COMMAND procedure, it would result in the Turtle jumping from whatever its position on the screen to the top of the box. It would be more meaningful if the child could move the Turtle to the top of the box using a sequence of commands. Therefore, a source of commands besides the computer is needed. A deck of picture cards like the one below can offer such a source. (See appendix for more examples.)



The child draws a card and tries to manipulate the Turtle to a position on the screen that matches the picture on the card. New cards can be added whenever a new spatial relation has been introduced to the child.

Comparisons

Another kind of spatial relation that might be interesting for children to work with on the computer is comparisons such as LONG, LONGER, LONGEST or LARGE, LARGER, LARGEST. The nature of this program will be somewhat different since the Turtle will be drawing on the screen. Although the format will be much the same as what we have been using so far, we are going to start a new program. The STOPs are added in this procedure.



```
TO COMPARE
MAKE "COM READWORD
IF  :COM = "LONG [PU SETPOS [-100 20] SETH 90 PD FD 50 PU] STOP
IF  :COM = "LONGER [SETPOS [-100 0] SETH 90 PD FD 100 PU] STOP
IF  :COM = "LONGEST [SETPOS [-100 -20] SETH 90 PD FD 150 PU] STOP
IF  :COM = "LARGE [CIRCLE1] STOP
IF  :COM = "LARGER [CIRCLE2] STOP
IF  :COM = "LARGEST [CIRCLE3] STOP
IF  :COM = "CLEAR [CLEARSCREEN] STOP
COMPARE
END
```

```
TO COMPARE
MAKE "COM READWORD
IF  :COM = "LONG PU SETXY - 100 20 SETH 90 PD FD 50 PU STOP
IF  :COM = "LONGER SETXY - 100 0 SETH 90 PD FD 100 PU STOP
IF  :COM = "LONGEST SETXY - 100 (- 20) SETH 90 PD FD 150 PU STOP
IF  :COM = "LARGE CIRCLE1 STOP
IF  :COM = "LARGER CIRCLE2 STOP
IF  :COM = "LARGEST CIRCLE3 STOP
IF  :COM = "CLEAR CLEARSCREEN STOP
END
```

```
TO CIRCLE1
PU SETPOS [-50 20] SETH 0 PD
REPEAT 120 [FD 1 RT 3]
PU
END
```

```
TO CIRCLE1
PU SETXY - 50 20 SETH 0 PD
REPEAT 120 [FD 1 RT 3]
PU
END
```

```
TO CIRCLE2
PU SETPOS [-50 20] SETH 0 PD
REPEAT 180 [FD 1 RT 2]
PU
END
```

```
TO CIRCLE2
PU SETXY - 50 20 SETH 0 PD
REPEAT 180 [FD 1 RT 2]
PU
END
```

```
TO CIRCLE3
PU SETPOS [-50 20] SETH 0 PD
REPEAT 360 [FD 1 RT 1]
PU
END
```

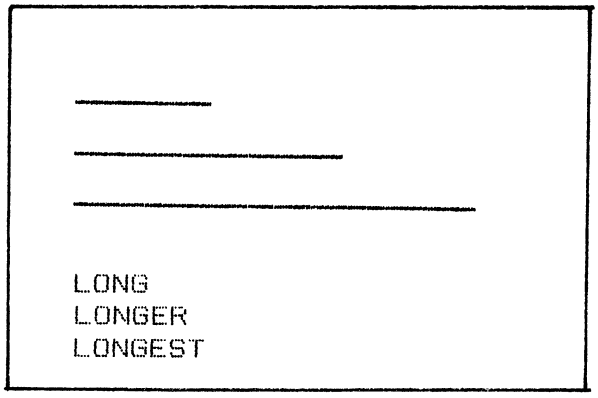
```
TO CIRCLE3
PU SETXY - 50 20 SETH 0 PD
REPEAT 360 [FD 1 RT 1]
PU
END
```

Since READWORD is not already defined in MIT Logo, it is necessary to define it.

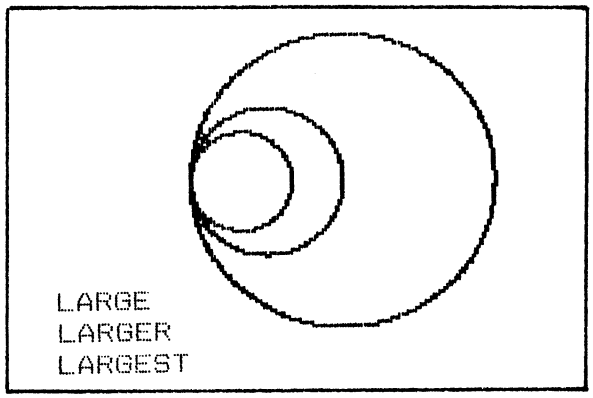
TO READWORD
OUTPUT FIRST REQUEST
END

Since the Turtle does not continue moving as in GETGOING, there is no need for that procedure. Instead COMPARE itself has been made recursive. The procedures for LONG, LONGER, and LONGEST immediately follow the commands since they are brief. The procedures for LARGE, LARGER, and LARGEST are more lengthy and so have been placed in the subprocedures CIRCLE1, CIRCLE2, and CIRCLE3.

If the child typed in LONG followed by LONGER and then LONGEST, the screen would look like the picture below.



Before typing further commands, the screen needs to be cleared. LARGE, LARGER, and LARGEST would result in a picture of three different size circles.



For the sake of variety and the experience of different geometric shapes, squares could be used for SMALL, SMALLER, and SMALLEST.

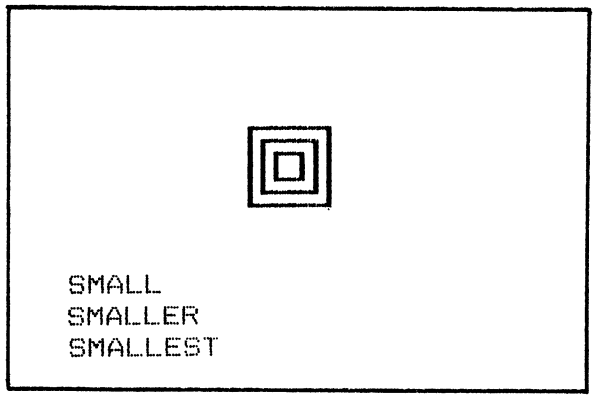
IF :COM = "SMALL [BOX1]
IF :COM = "SMALLER [BOX2]
IF :COM = "SMALLEST [BOX3]

IF :COM = "SMALL BOX1
IF :COM = "SMALLER BOX2
IF :COM = "SMALLEST BOX3

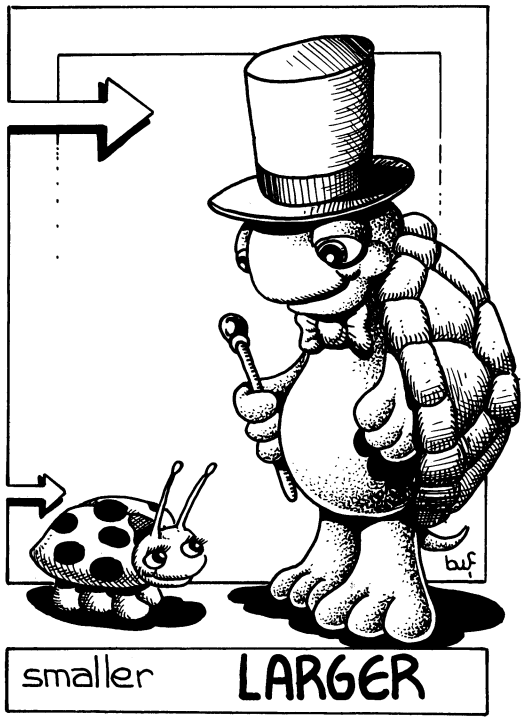
```
TO BOX1
PU SETPOS [-15 0] SETH 0 PD
REPEAT 4 [FD 30 RT 90] PU
END

TO BOX2
PU SETPOS [-10 5] SETH 0 PD
REPEAT 4 [FD 20 RT 90] PU
END

TO BOX3
PU SETPOS [-5 10] SETH 0 PD
REPEAT 4 [FD 10 RT 90] PU
END
```

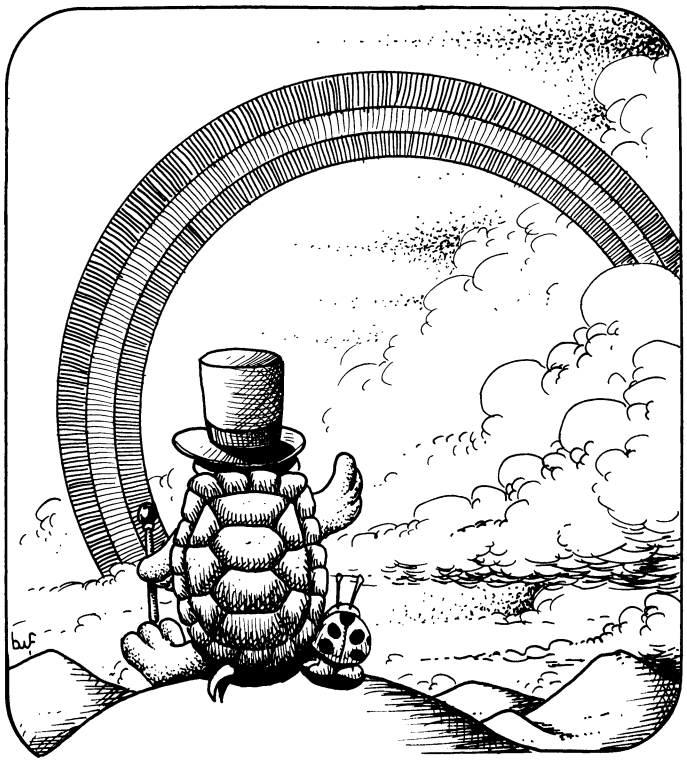
```
TO BOX1
SETXY - 15 0 SETH 0 PD
REPEAT 4 [FD 30 RT 90] PU
END

TO BOX2
SETXY - 10 5 SETH 0 PD
REPEAT 4 [FD 20 RT 90] PU
END

TO BOX3
SETXY - 5 10 SETH 0 PD
REPEAT 4 [FD 10 RT 90] PU
END
```

The screen would show successively smaller squares, one inside the other.



Any number of comparisons can be added to the program. However, as more and more comparisons are added, remember the program may become congested and begin to slow down. If this happens, it is necessary to put COMPARE inside another recursive program before putting a STOP after each of the commands. The program in the appendix is such a program and is called COMPARISONS.





Colors

If you have a color monitor, you might want to use a program that will allow the children to learn to spell the different colors. It's a simple program to write. We'll use the same basic program that we have been using, but now we'll call it COLORS.

```
TO COLORS
SPLITSCREEN HT
MAKE "COM READKEY
IF :COM = "BLUE [SETPC 5 FILL]
IF :COM = "GREEN [SETPC 2 FILL]
IF :COM = "WHITE [SETPC 1 FILL]
IF :COM = "PURPLE [SETPC 3 FILL]
IF :COM = "ORANGE [SETPC 4 FILL]
COLORS
END
```

```
TO COLORS
SPLITSCREEN PU HOME PD HT
MAKE "COM READKEY
IF :COM = "BLUE PC 5 FILL
IF :COM = "GREEN PC 2 FILL
IF :COM = "WHITE PC 1 FILL
IF :COM = "PURPLE PC 3 FILL
IF :COM = "ORANGE PC 4 FILL
COLORS
END
```

```
TO READKEY
IF KEYP [CLEARTEXT OUTPUT READWORD]
OUTPUT "
END
```

```
TO READKEY
IF RC? CLEARTEXT OUTPUT FIRST REQUEST
OUTPUT "
END
```

```
TO FILL
CLEARSCREEN
REPEAT 4 [FD 20 RT 90]
REPEAT 10 [FD 20 RT 90 FD 1 RT 90 FD 20 LT 90 FD 1 LT 90]
END
```

SETPC or *PC* is the command for setting the pencolor. The number 5 corresponds to blue, 2 to green, 1 to white, etc. Apple Logo and MIT Logo for the Apple have only six colors and so are limited in what can be done with this program. Most of the other versions of Logo have more colors from which to choose. READKEY is the same tool procedure as used in GETGOING. FILL is basically the SQUARE procedure in that same program.

Arithmetic

One of the important things that we do with children in kindergarten and the early grades is to cultivate a concept of numbers. We begin by counting objects and then assigning a number to the group. Then we teach the children how to combine groups and how to take things away from a group. It is important that children spend a lot of time moving objects around when they are learning to count, to add, and to subtract.

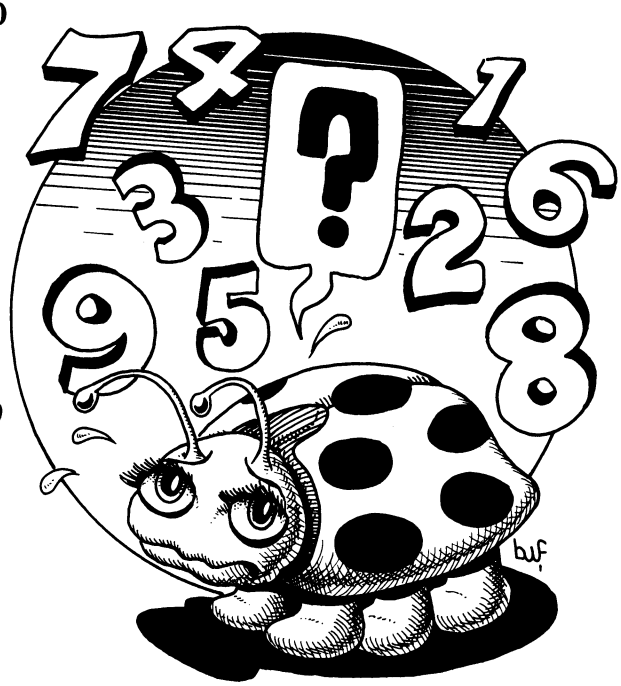
After children have accumulated enough experience manipulating concrete objects such as blocks, beads, or beans, they develop a readiness for abstraction. They can abstract fiveness from a group of five beans, a group of five blocks, and a group of five beads. Fiveness is one of the attributes that all three groups have in common.

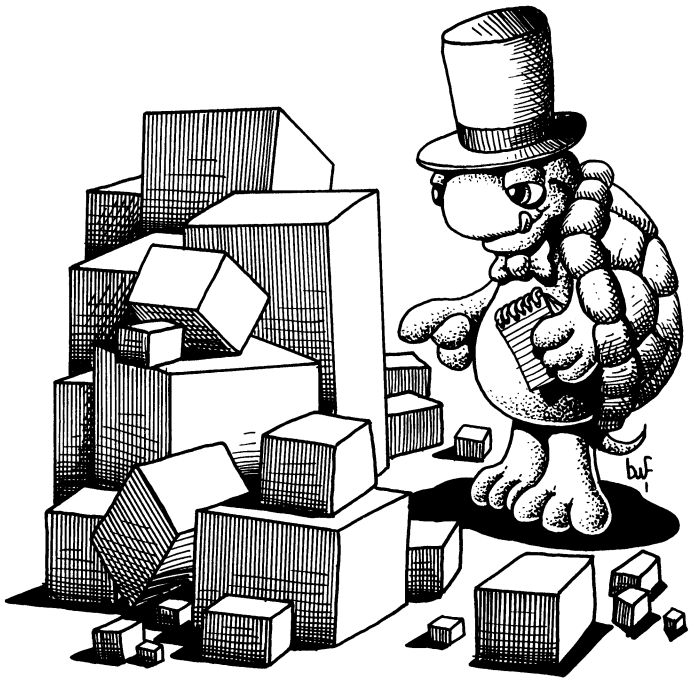
The process of abstraction is a very subtle and complex process. Meaning accumulates with experience and, if hurried, can be lost entirely. The COUNTING, ADDING, and SUBTRACTING programs are designed to help children make an easier transition from the concrete to the abstract. The 3-dimensional blocks become 2-dimensional screen boxes. Touch is preserved to some extent through the use of the keyboard. The child hits a key to add, subtract, and count boxes. In using this program, children tend to touch the screen a lot as they count the boxes. It is important to allow them to do so if they need that one-to-one correspondence of finger touching box. Through the keyboard and fingering of the screen some concreteness is preserved. The program is more abstract than the physical handling of objects, but less abstract than the manipulation of symbols like 5 + 7 or 9 – 4.

Since the three programs are very much alike, COUNTING can be easily modified to do ADDING or SUBTRACTING.

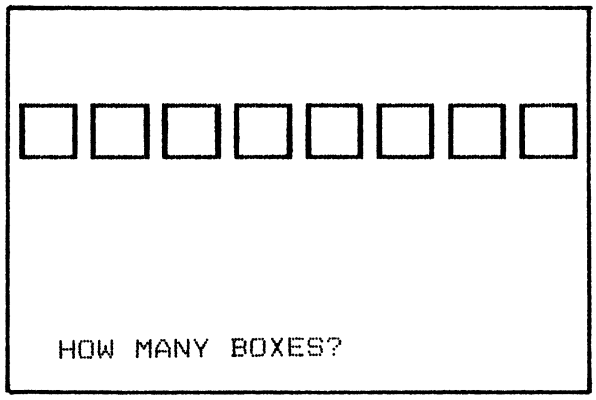
```
TO COUNTING
HT CS CLEARTEXT
MAKE "NUMBER 1 + RANDOM 9
PU SETPOS [-130 60]
BOXES :NUMBER -130
GETANSWER
IF NOT : DIDITRIGHT [STOP]
PRINT [TRY ANOTHER ONE.] WAIT 90
COUNTING
END
```

```
TO COUNTING
HT CS CLEARTEXT
MAKE "NUMBER 1 + RANDOM 9
PU SETXY - 130 60
BOXES :NUMBER ( - 130 )
GETANSWER
IF NOT :DIDITRIGHT STOP
PRINT [TRY ANOTHER ONE.] WAIT 90
COUNTING
END
```





The first thing we want our procedure to do is to put a random number of boxes on the screen in the position that we designate. The MAKE command assigns a number between 1 and 10 to the variable called NUMBER. The procedure BOXES then draws that particular NUMBER of boxes starting at an X-coordinate of −130 and a Y-coordinate of 60.




```
TO BOXES :NUMBER :X
REPEAT :NUMBER [SETX :X PD BOX PU MAKE "X :X + 27] SETX -130
END

TO BOX
REPEAT 4 [FD 20 RT 90]
END
```

BOXES uses a MAKE command to establish the X-coordinate for each new box (MAKE "X :X + 27). Each BOX is 20 turtlesteps on a side with 7 turtlesteps between boxes.

We now want the program to ask the child how many boxes are on the screen and to check to see if his answer is correct.

```
TO GETANSWER
CLEARTEXT
PRINT [HOW MANY BOXES?]
MAKE "ANSWER READNUMBER
TEST :ANSWER = :NUMBER
IFTRUE [PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE]
IFFALSE [CLEARTEXT PRINT [HIT THE "C" ONE TIME FOR EACH BOX.]
  MAKE "KEEPTRACK 0 OPERATIONS CHECK]
END
```

```
TO GETANSWER
CLEARTEXT
PRINT [HOW MANY BOXES?]
MAKE "ANSWER READNUMBER
TEST  :ANSWER =  :NUMBER
IFTRUE PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE
IFFALSE CLEARTEXT PRINT [HIT THE "C" ONE TIME FOR EACH BOX.] MAKE
  "KEEPTRACK 0 OPERATIONS CHECK
END
```

READNUMBER is a tool procedure that determines whether or not the child has entered a number. It outputs the number to GETANSWER which tests to see if the number is the correct answer. READNUMBER and tool procedures like it can be found in most of the reference manuals on Logo. This particular version was taken from Dan Watt's **Learning With Logo**, an excellent reference.

```
TO READNUMBER
MAKE "ANSWER READLIST
TEST :ANSWER = [ ]
IFTRUE [PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER]
TEST NOT NUMBERP FIRST :ANSWER
IFTRUE [PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER]
IFFALSE [OUTPUT FIRST :ANSWER]
END
```

```
TO READNUMBER
MAKE "ANSWER REQUEST
TEST  :ANSWER = [ ]
IFTRUE PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER
TEST NOT NUMBER? FIRST  :ANSWER
IFTRUE PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER
IFFALSE OUTPUT FIRST  :ANSWER
END
```

If the child types in the correct number of boxes, control is returned to COUNTING and he is given another screen of boxes to count. If, however, his answer is not correct, he is instructed to "Hit the 'C' key once for each box" and a procedure named OPERATIONS is called. OPERATIONS (including READKEY) allows the child to use the "C" key to FILL in the boxes while keeping track of the number filled. When every box on the screen has been filled (IF :KEEPTRACK = :NUMBER), OPERATIONS stops and CHECK is called.

```
TO OPERATIONS
IF :KEEPTRACK = :NUMBER [STOP]
MAKE "OP READKEY
IF :OP = "C [FILL MAKE "KEEPTRACK :KEEPTRACK + 1]
OPERATIONS
END

TO READKEY
IF KEYP [OUTPUT READCHAR]
OUTPUT "
END
```

```
TO OPERATIONS
IF  :KEEPTRACK =  :NUMBER STOP
MAKE "OP READKEY
IF  :OP = "C FILL MAKE "KEEPTRACK  :KEEPTRACK + 1
OPERATIONS
END

TO READKEY
IF RC? OUTPUT READCHARACTER
OUTPUT "
END
```

```
TO FILL
PD REPEAT 10[FD 20 RT 90 FD 1 RT 90 FD 20 LT 90 FD 1 LT 90] PU
RT 90 FD 7 LT 90
END
```

CHECK determines if the child knows how many boxes were filled. If he does, thereby making DIDITRIGHT true, control is returned to COUNTING, and the child is given another try. If the child does not know the answer (making DIDITRIGHT false) control is returned to COUNTING and the child must talk with the teacher.

```
TO CHECK
CLEARTEXT PRINT [HOW MANY BOXES WERE FILLED?]
MAKE "ANSWER READNUMBER
TEST :ANSWER = :NUMBER
IFTRUE [CLEARTEXT PRINT [GOOD! MAKE "DIDITRIGHT "TRUE]
IFFALSE [CLEARTEXT PRINT [TALK WITH YOUR TEACHER.]
  MAKE "DIDITRIGHT "FALSE
END
```

```
TO CHECK
CLEARTEXT PRINT [HOW MANY BOXES WERE FILLED?]
MAKE "ANSWER READNUMBER
TEST  :ANSWER =  :NUMBER
IFTRUE CLEARTEXT PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE
IFFALSE CLEARTEXT PRINT [TALK WITH YOUR TEACHER.] MAKE "DIDITRIGHT
  "FALSE
END
```

WAIT must be defined for MIT Logo.

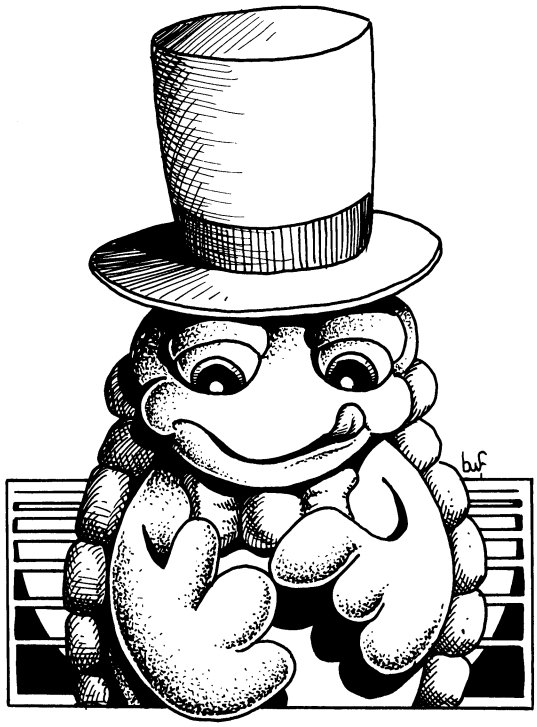
```
TO WAIT  :TIME
IF  :TIME = 0 STOP
WAIT  :TIME - 1
END
```

Since COUNTING is a recursive procedure, it will continue indefinitely unless broken by CTRL-G or unless the child misses the correct answer after filling the boxes and the procedure stops.

The COUNTING procedure is one that is easy to change to a procedure that adds or subtracts. If it's ADDING that you want, change the name of the procedure accordingly. Then it will be necessary to write a procedure that will ADD a second row of boxes.

```
TO ADDING
HT CS CLEARTEXT
MAKE "NUMBER 1 + RANDOM 9
MAKE "NUMBER2 1 + RANDOM 9
PU SETPOS [-130 60]
BOXES :NUMBER -130
ADD
GETANSWER
IF NOT : DIDITRIGHT [STOP]
PRINT [TRY ANOTHER ONE.] WAIT 90
ADDING
END
```

```
TO ADDING
HT CS CLEARTEXT
MAKE "NUMBER 1 + RANDOM 9
MAKE "NUMBER2 1 + RANDOM 9
PU SETXY - 130 60
BOXES  :NUMBER ( - 130 )
ADD
GETANSWER
IF NOT  :DIDITRIGHT STOP
PRINT [TRY ANOTHER ONE.] WAIT 90
ADDING
END
```



```
TO ADD
PRINT (SENTENCE [HIT THE "A" KEY] :NUMBER2 [TIMES.])
MAKE "KEEPTRACK 0
OPERATIONS
END
```

ADDING requires two random numbers. The first row of boxes is put up as in COUNTING. ADD instructs the child to hit the "A" key a number of times corresponding to the second random number. OPERATIONS now draws a second row of boxes (BOXES2) instead of filling in the boxes as in COUNTING.

```
TO OPERATIONS
IF :KEEPTRACK = :NUMBER2 [STOP]
MAKE "OP READKEY
IF :OP = "A [BOXES2 MAKE "KEEPTRACK :KEEPTRACK + 1]
OPERATIONS
END
```

```
TO OPERATIONS
IF  :KEEPTRACK =  :NUMBER2 STOP
MAKE "OP READKEY
IF  :OP = "A BOXES2 MAKE "KEEPTRACK  :KEEPTRACK + 1
OPERATIONS
END
```

```
TO BOXES2
PU SETY 0 PD
BOX
PU RT 90 FD 27 LT 90 PD
END
```

GETANSWER is changed slightly. The answer is tested to see if it equals the sum of the two numbers (TEST :ANSWER = :NUMBER + :NUMBER2). If the test is true, then control is returned to ADDING. If it is false, the procedure calls CHECK. CHECK, also, is only slightly changed.

```
TO GETANSWER
CLEARTEXT
PRINT [HOW MANY BOXES?]
MAKE "ANSWER READNUMBER
TEST  :ANSWER =  :NUMBER +  :NUMBER2
IFTRUE [PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE]
IFFALSE [CHECK]
END
```

```
TO GETANSWER
CLEARTEXT
PRINT [HOW MANY BOXES?]
MAKE "ANSWER READNUMBER
TEST  :ANSWER =  :NUMBER +  :NUMBER2
IFTRUE PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE
IFFALSE CHECK
END
```

```
TO CHECK
CLEARTEXT
PRINT [COUNT THE BOXES AGAIN.]
PRINT [TYPE THE NUMBER.]
MAKE "ANSWER READNUMBER
TEST :ANSWER = :NUMBER + :NUMBER2
IFTRUE [CLEARTEXT PRINT [GOOD!]MAKE DIDITRIGHT "TRUE]
IFFALSE [CLEARTEXT PRINT [TALK WITH YOUR TEACHER.]
  MAKE "DIDITRIGHT "FALSE]
END
```

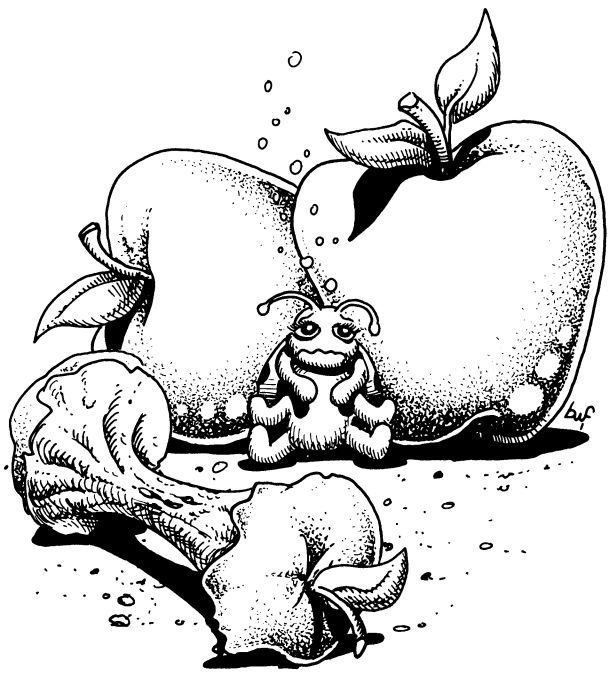
```
TO CHECK
CLEARTEXT
PRINT [COUNT THE BOXES AGAIN.]
PRINT [TYPE THE NUMBER.]
MAKE "ANSWER READNUMBER
TEST  :ANSWER =  :NUMBER +  :NUMBER2
IFTRUE CLEARTEXT PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE
IFFALSE CLEARTEXT PRINT [TALK WITH YOUR TEACHER.] MAKE "DIDITRIGHT
  "FALSE
END
```

If the child's answer is correct on the second try, control returns to ADDING. If not, the child is referred to the teacher, control returns to ADDING and the procedure stops. The tool procedures, READKEY and READNUMBER, remain unchanged as do the procedures for drawing the first row of boxes (BOXES and BOX).

The SUBTRACTING procedure results from similar modifications. It chooses two numbers which must be compared to determine if NUMBER2 is larger than NUMBER. if so, another random number must be generated.

```
TO SUBTRACTING
HT CS CLEARTEXT
CHOOSENUMBERS
PU SETPOS [-130 60]
BOXES :NUMBER -130
SUBTRACT
GETANSWER
IF NOT  :DIDITRIGHT [STOP]
PRINT [TRY ANOTHER ONE.] WAIT 90
SUBTRACTING
END
```

```
TO SUBTRACTING
HT CS CLEARTEXT
CHOOSENUMBERS
PU SETXY - 130 60
BOXES  :NUMBER ( - 130 )
SUBTRACT
GETANSWER
IF NOT  :DIDITRIGHT STOP
PRINT [TRY ANOTHER ONE.] WAIT 90
SUBTRACTING
END
```



```
TO CHOOSENUMBERS
MAKE "NUMBER 1 + RANDOM 9
MAKE "NUMBER2 1 + RANDOM 9
IF :NUMBER2 > :NUMBER [CHOOSENUMBERS]
END
```

```
TO CHOOSENUMBERS
MAKE "NUMBER 1 + RANDOM 9
MAKE "NUMBER2 1 + RANDOM 9
IF :NUMBER2 > :NUMBER CHOOSENUMBERS
END
```

SUBTRACT instructs the child to use the "S" key to ERASE.BOXES.

```
TO SUBTRACT
PRINT (SE [HIT THE "S" KEY] :NUMBER2 [TIMES.])
MAKE "KEEPTRACK 0
OPERATIONS
END
```

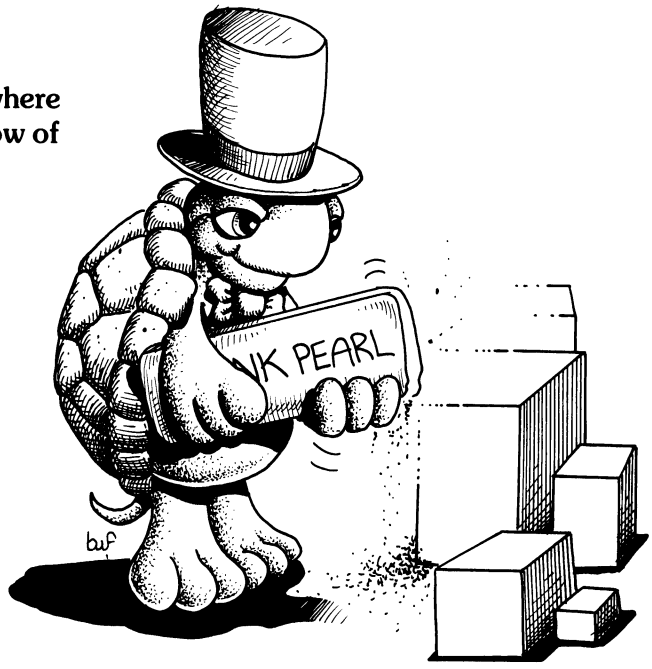
The only change in OPERATIONS is in terms of the key that it reads and the procedure that follows it.

```
IF :OP = "S [ERASE.BOXES MAKE "KEEPTRACK :KEEPTRACK + 1]
```

```
TO OPERATIONS
IF :KEEPTRACK = :NUMBER2 [STOP]
MAKE "OP READKEY
IF :OP = "S [ERASE.BOXES MAKE "KEEPTRACK :KEEPTRACK + 1]
OPERATIONS
END
```

```
TO OPERATIONS
IF :KEEPTRACK = :NUMBER2 STOP
MAKE "OP READKEY
IF :OP = "S ERASE.BOXES MAKE "KEEPTRACK :KEEPTRACK + 1
OPERATIONS
END
```

ERASE.BOXES starts from the position where the Turtle stopped after drawing the first row of boxes.




```
TO ERASE.BOXES
PE BOX
PU LT 90 FD 7 FD 20 RT 90
END
```

```
TO ERASE.BOXES
PC 0 BOX PC 1
PU LT 90 FD 7 FD 20 RT 90
END
```

GETANSWER and CHECK differ only in the test line.

```
TEST :ANSWER = :NUMBER - :NUMBER2
```

The complete listing of all procedures can be found in Appendix B.

COUNTING, ADDING, and SUBTRACTING can be modified in a number of ways. A particular number of problems can be assigned by using a variable with the procedure, such as COUNTING :N. (Remember to include COUNTING :N - 1 within the procedure.) A conditional statement would then stop the procedure when :N equaled 0.

```
IF :N = 0 [STOP]
```

IF : N = 0 STOP

ADDING and SUBTRACTING can print the symbols for the addition or subtraction instead of printing "How many boxes?" Simply change the PRINT statement in GETANSWER to read either of the following.

```
PRINT (SENTENCE :NUMBER [+] :NUMBER2 [=])
PRINT (SENTENCE :NUMBER [-] :NUMBER2 [=])
```

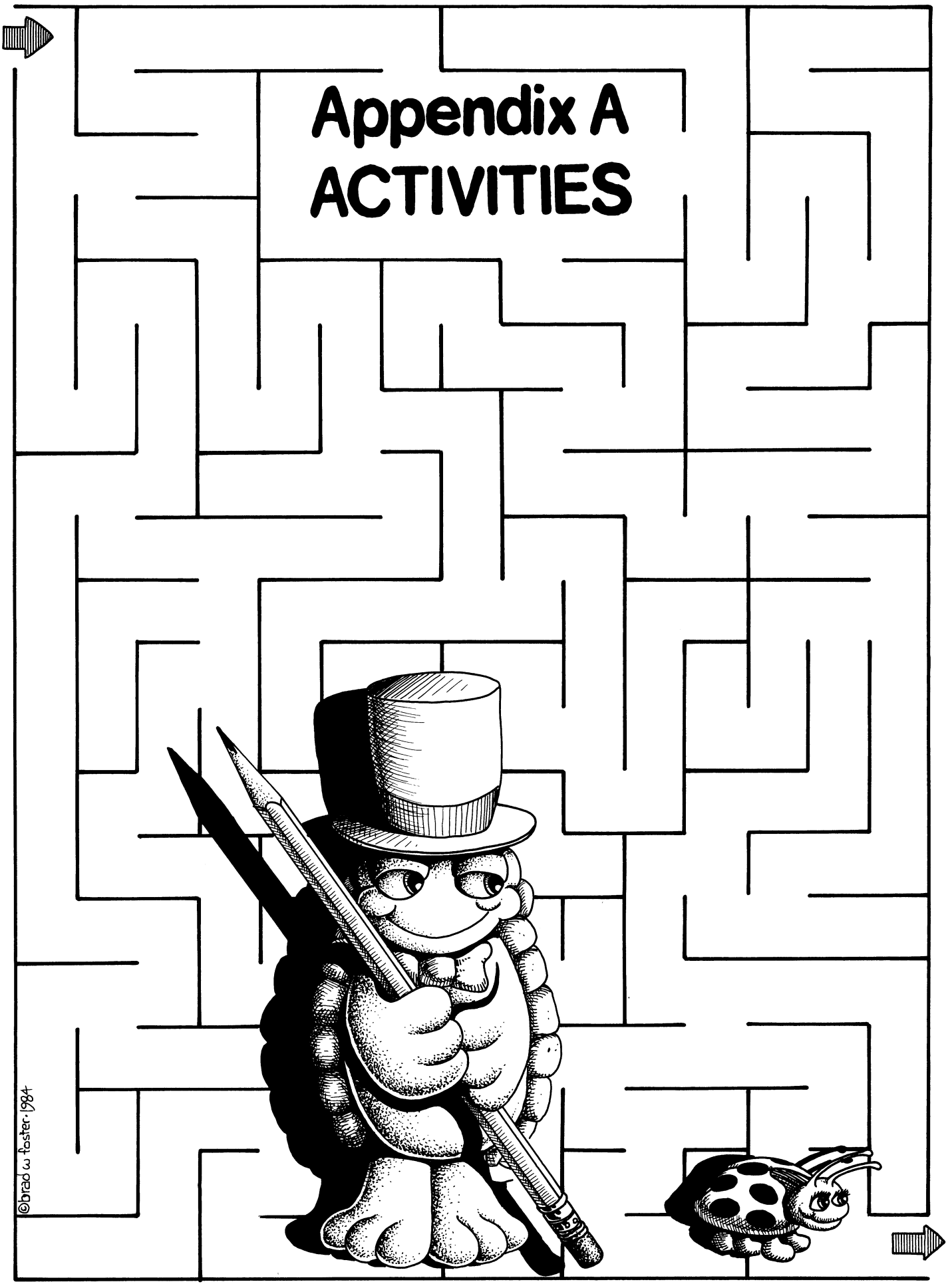
Rather than having the procedure print the symbolic representation for the addition or subtraction, it might be a good idea to have the child translate the objects to numbers. If he is ADDING and the screen shows 5 boxes, he would record a "5". If he is instructed to "Hit the 'A' key 6 times," he would record a "+6". In response to "How many boxes?" he would record "11". The child has then made a transition from concrete to abstract. All learning entails a movement back and forth between the concrete and the abstract. With children, it is important to choose activities which encourage such movement.

An Afterthought

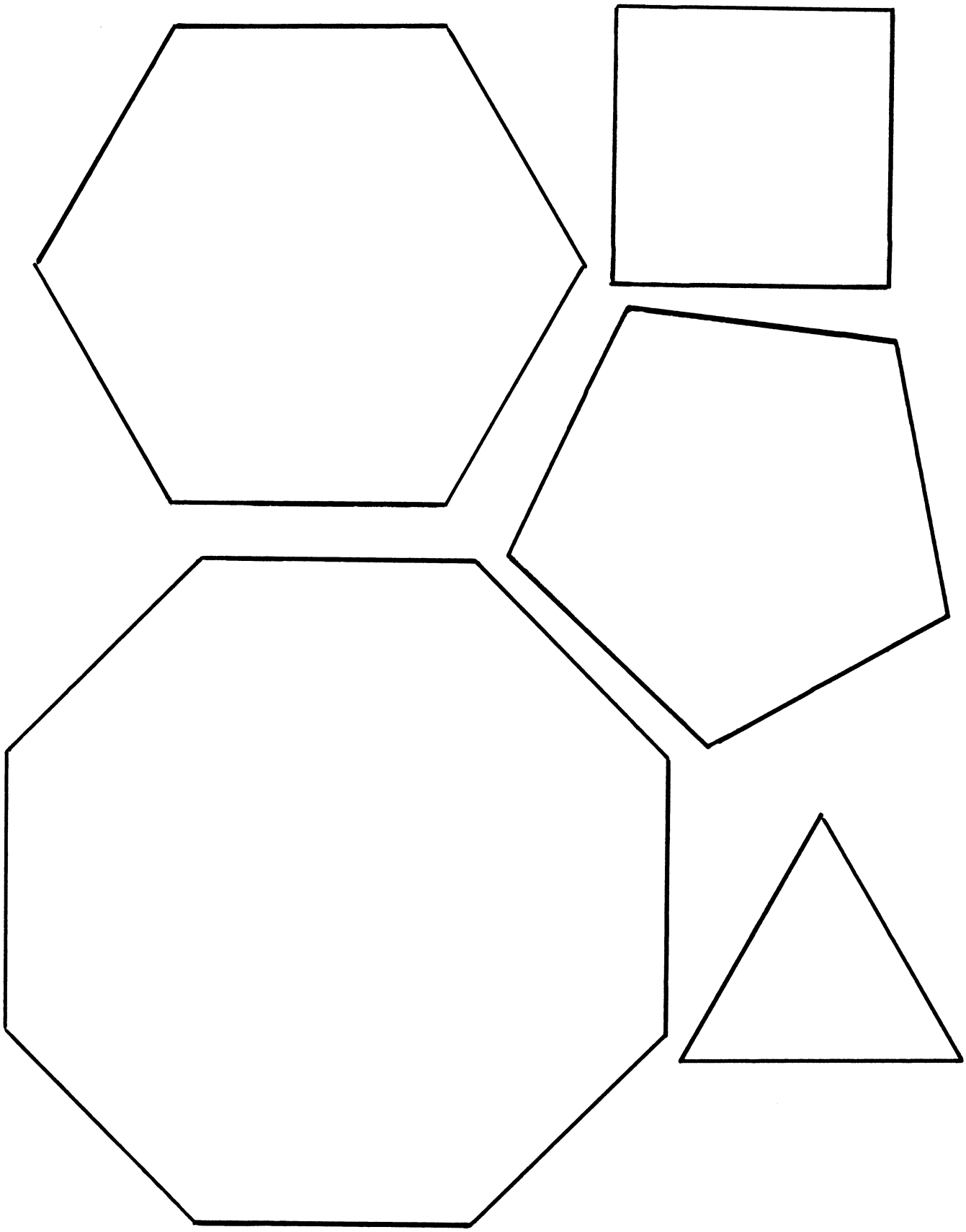
In working with Logo, it is important to keep in mind the things that all good teachers know about children and about learning. The educator and philosopher David Hawkins puts it well.

> "Children only learn well of their own accord. ... When children are only dragged, they learn to cope with a curriculum of intimidation. To teach, then, is to have the authority of the guide, of one who shows the way. To teach is not to drag, though some element of the imperative mood is always latent in teaching; the guide who shows the way is never merely permissive. ... Good teaching is above all a preparation for the unforeseen, for the lovely things that can happen when one has faith that they will happen."
>
> —David Hawkins. "Nature Closely Observed," Daedalus 112 (2) Spring, 1983.

Appendix A
ACTIVITIES

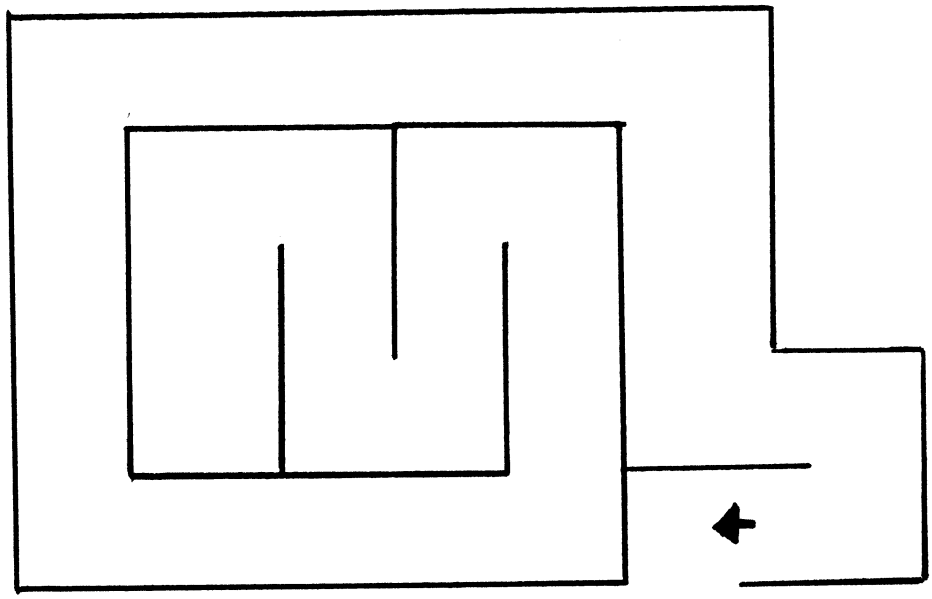


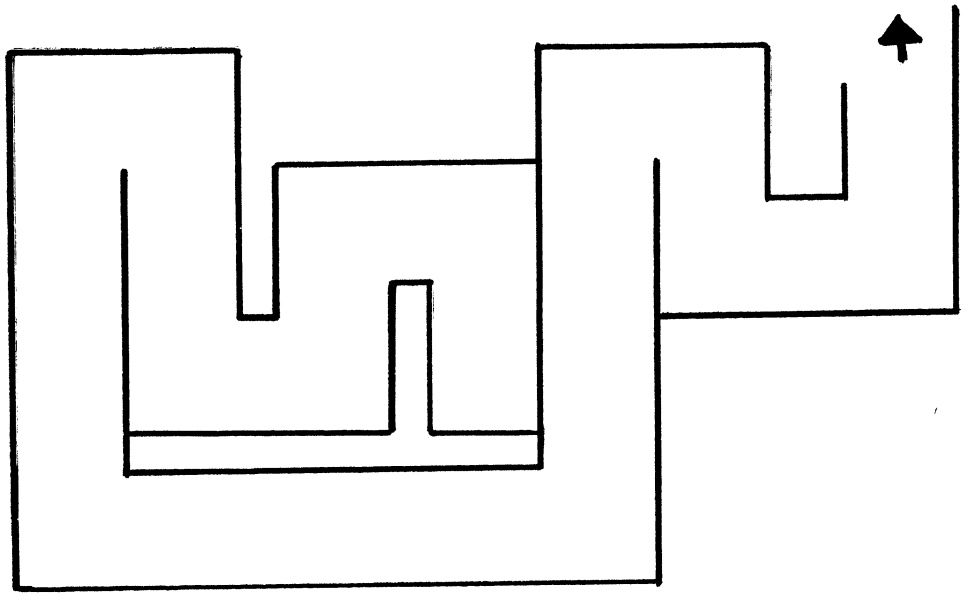
Polygon Patterns



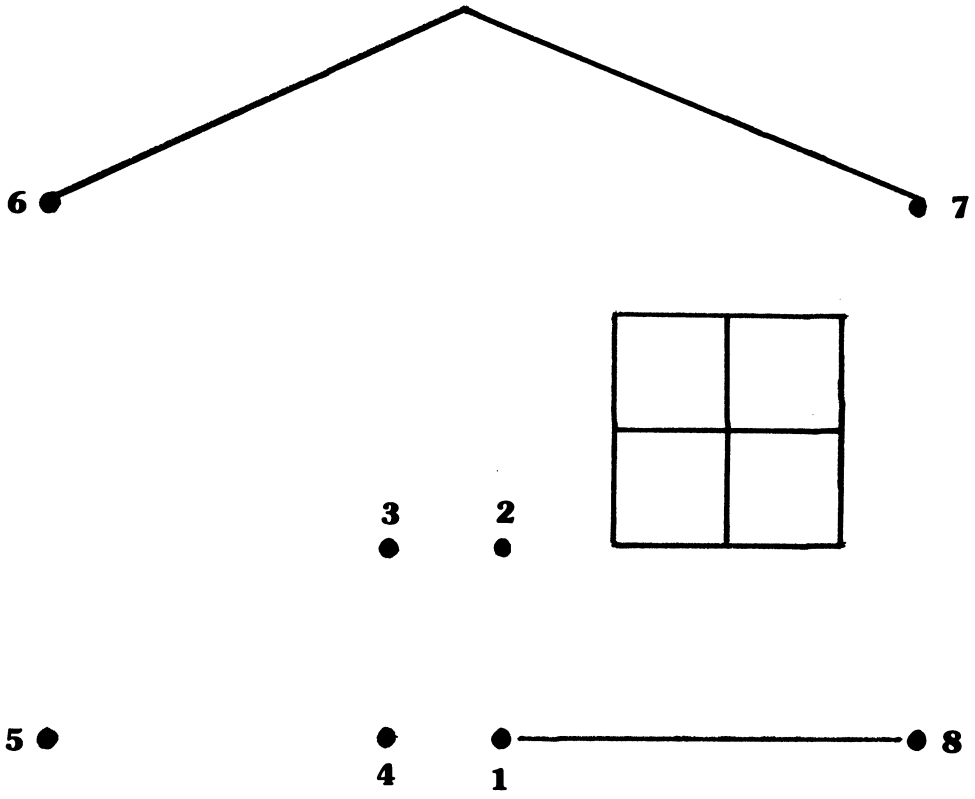
Mazes
Level I

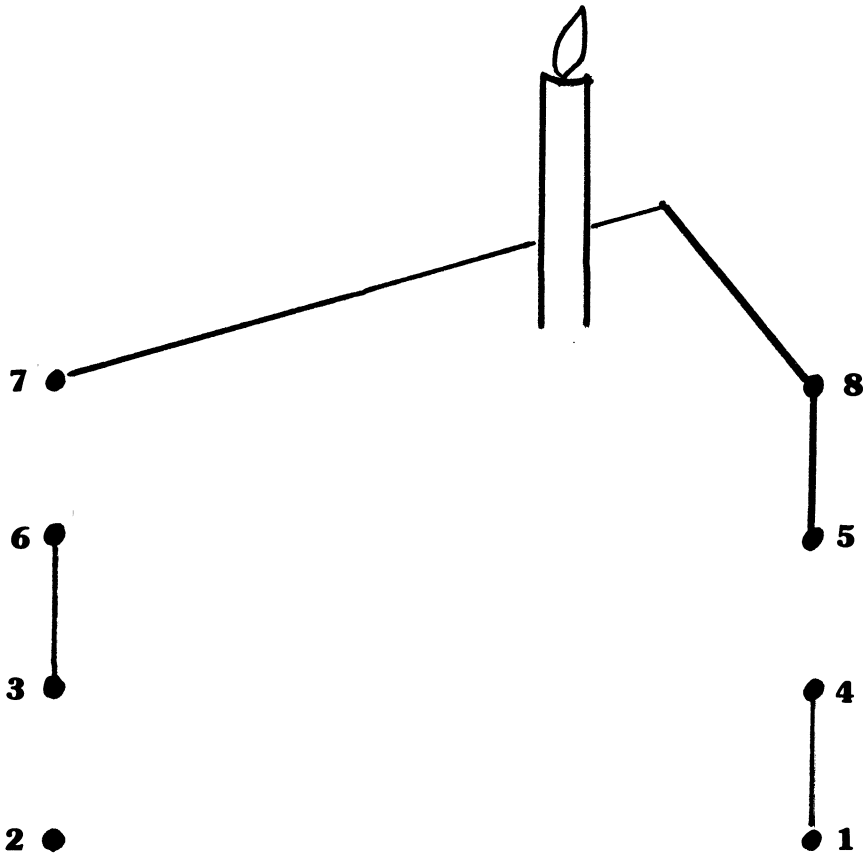
(Duplicate on overhead transparency film.)





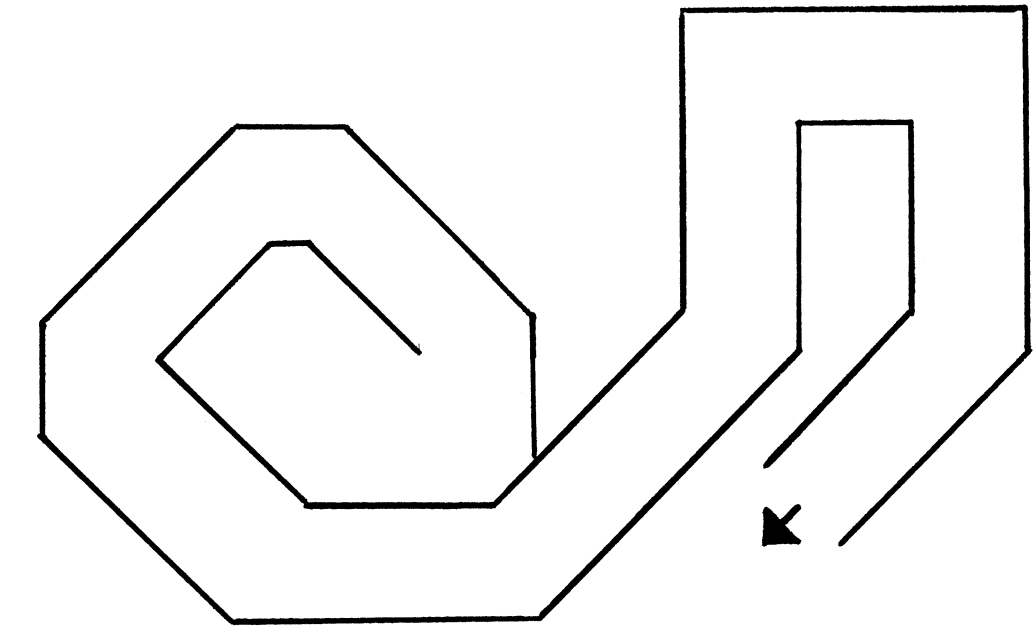
Dot-to-Dot
Level I

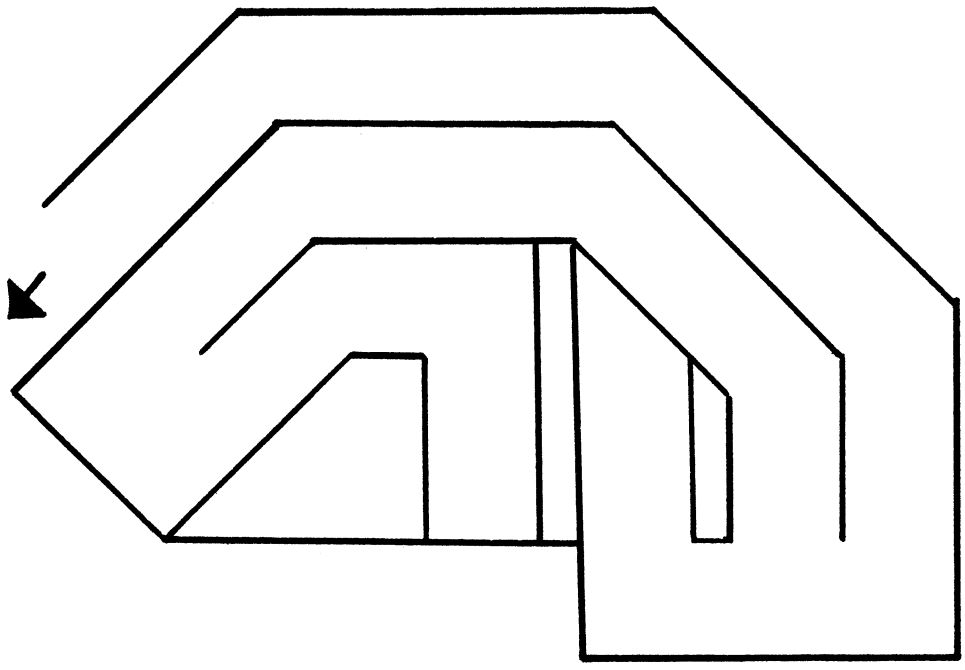




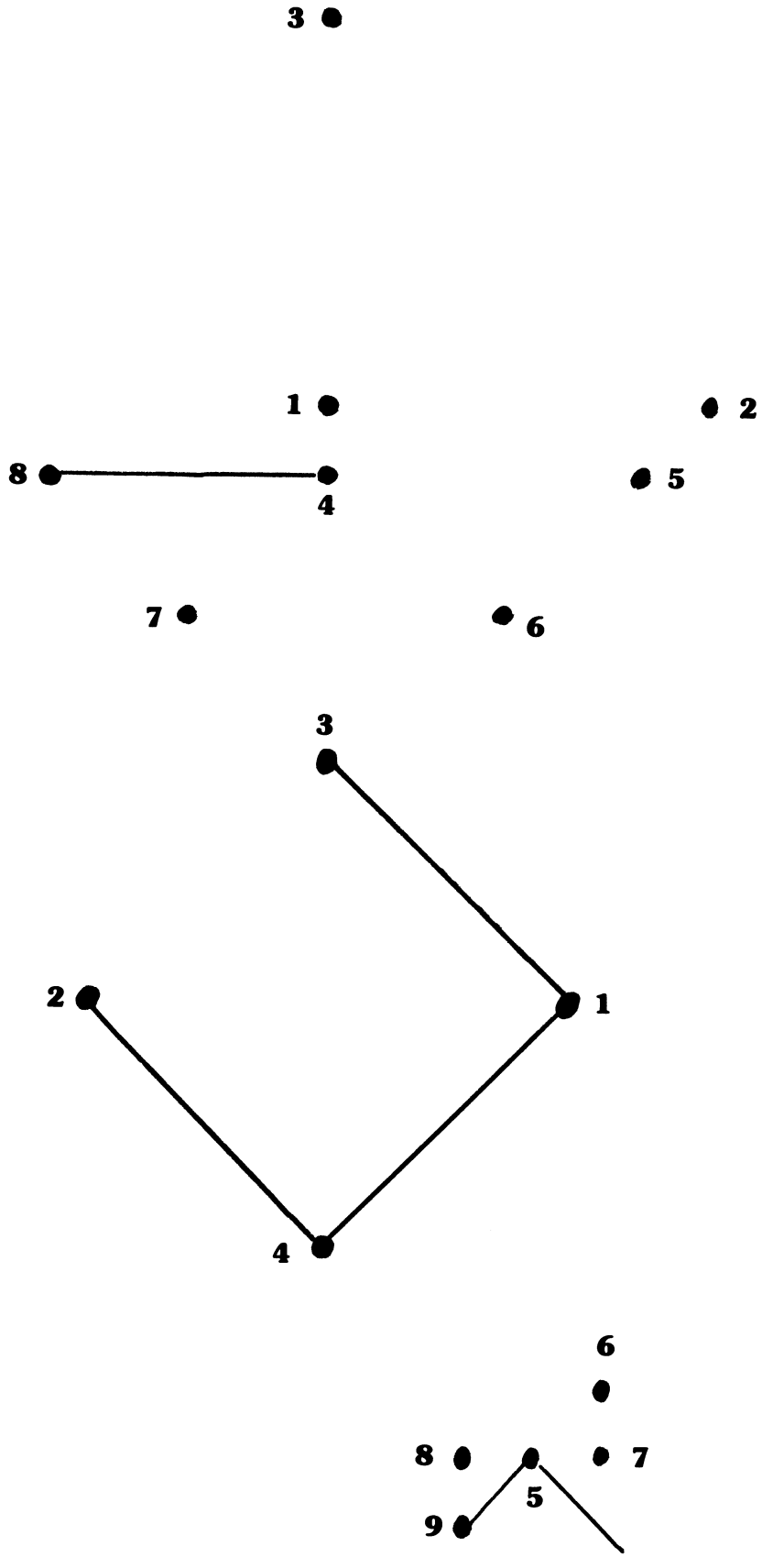
Mazes
Level II

(Duplicate on overhead transparency film.)





Dot-to-Dot
Level II



Graph Paper Pictures
Level 1

Dog

Forward 2
Left
Forward 1
Left
Forward 1
Right
Forward 1
Left
Forward 1
Left
Forward 6
Left
Forward 1
Back 3
Left
Forward 1
Back 1
Right
Forward 1
Left
Forward 4
Right
Forward 1
Back 2
Left
Forward 1

Cowboy

Left
Forward 3
Right
Forward 1
Right
Forward 1
Left
Forward 1
Right
Forward 2
Right
Forward 1
Left
Forward 1
Right
Forward 1
Right
Forward 1
Left
Forward 2
Left
Forward 3
Right
Forward 1
Right
Forward 3
Left
Forward 2
Left
Forward 1
Right
Forward 3
Left
Forward 1
Right
Forward 1
Right
Forward 2
Right
Forward 3
Left
Forward 2
Left
Forward 3
Right
Forward 2
Right
Forward 1
Right
Forward 1
Left
Forward 3
Right
Forward 1
Left
Forward 2
Left
Forward 3
Right
Forward 1
Right
Forward 3
Left
Forward 2

Snail

Forward 1
Right
Forward 1
Right
Forward 2
Right
Forward 2
Right
Forward 3
Right
Forward 3
Right
Forward 4
Right
Forward 4
Right
Forward 5
Right
Forward 5
Right
Forward 6
Right
Forward 7
Right
Forward 5
Left
Forward 2
Left
Forward 2
Left
Forward 2

Graph Paper Pictures
Level 2

Fish

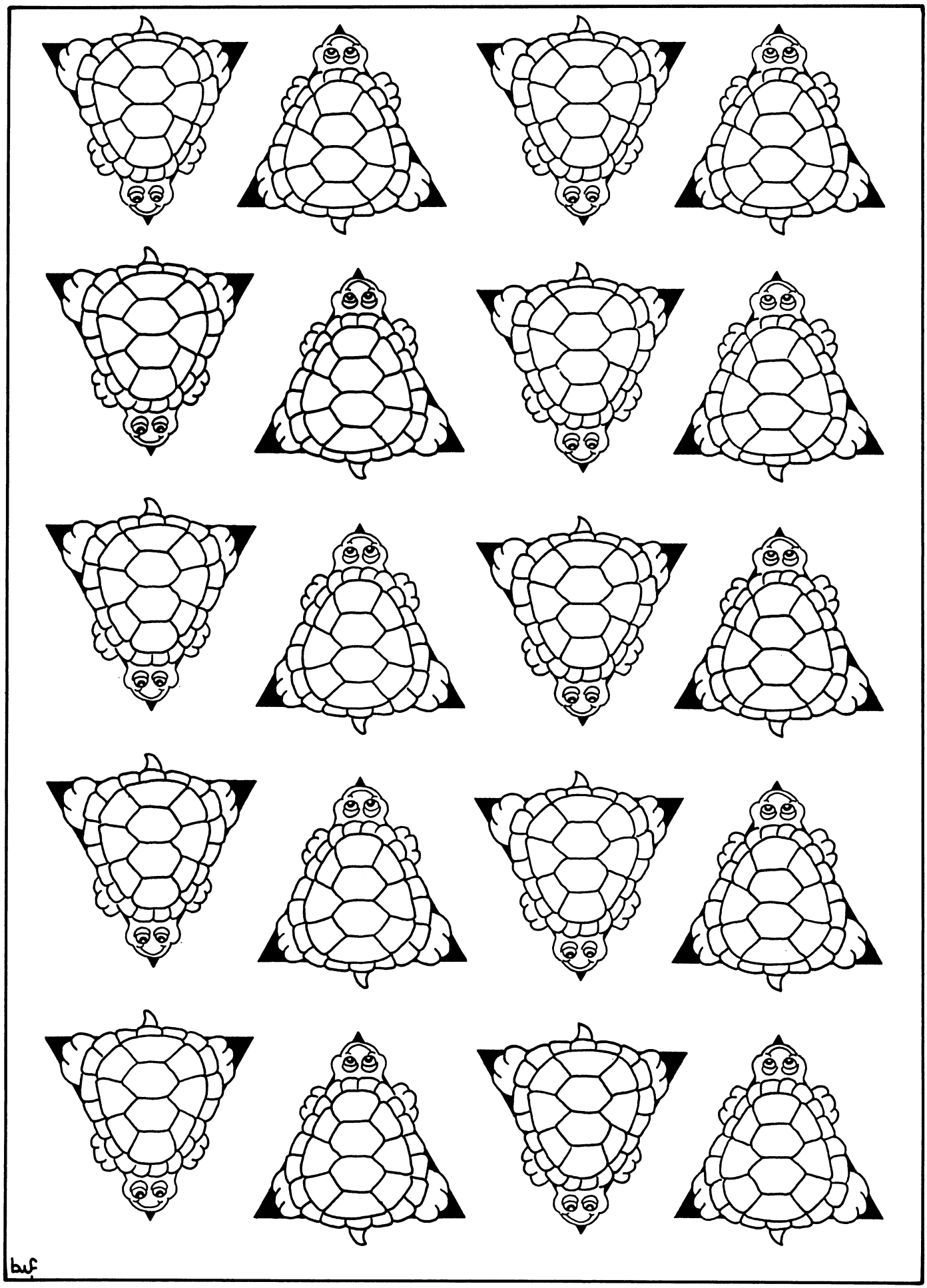
Forward 4
3 half-turns left
Forward 4
1 half-turn right
Forward 2
1 half-turn right
Forward 2
2 half-turns right
Forward 2
1 half-turn right
Forward 2
1 half-turn right
Forward 4

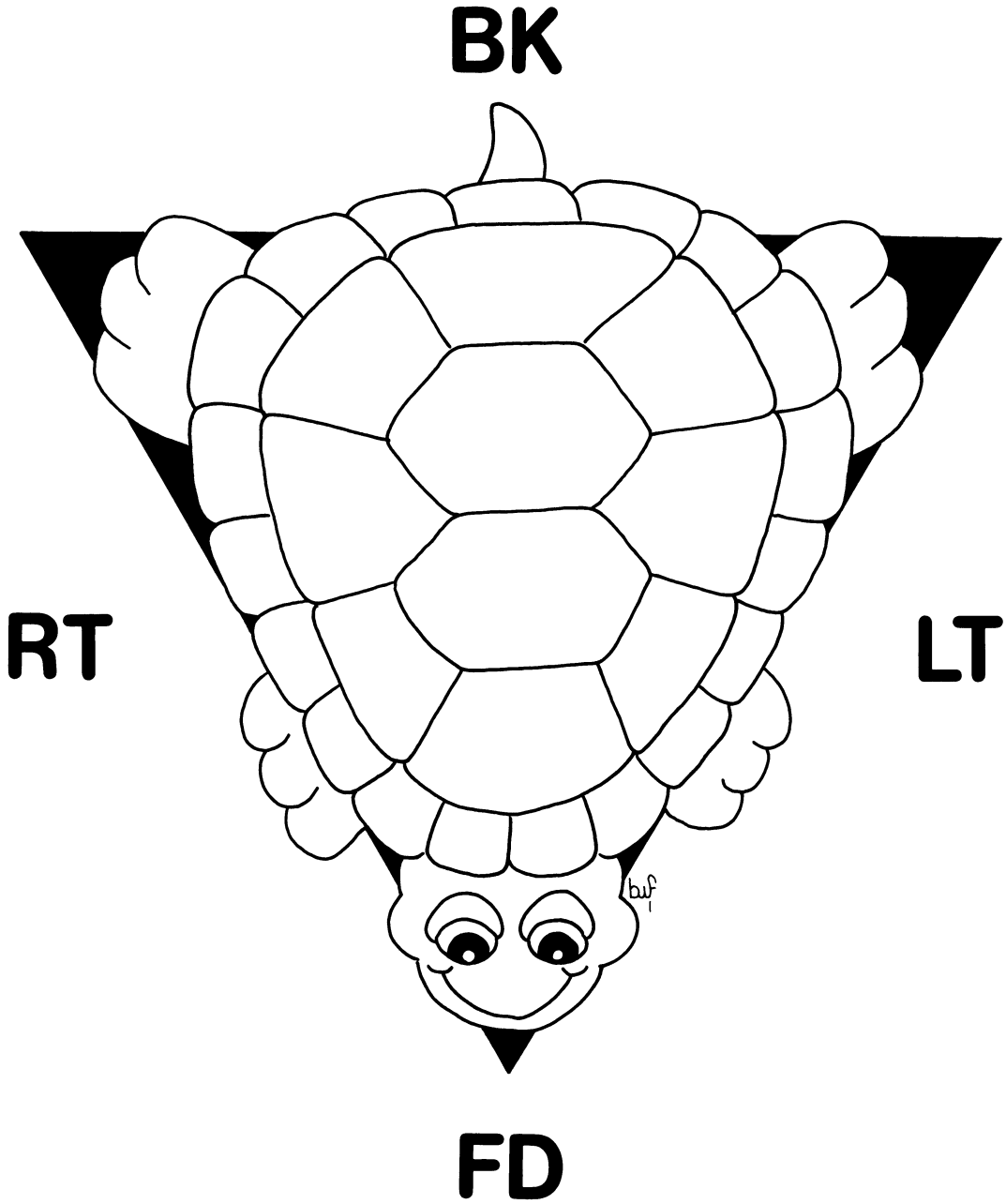
Heart

Forward 2
1 half-turn right
Forward 1
1 half-turn right
Forward 1
1 half-turn right
Forward 1
2 half-turns left
Forward 1
1 half-turn right
Forward 1
1 half-turn right
Foward 1
1 half-turn right
Forward 2
1 half-turn right
Forward 3
2 half-turns right
Forward 3

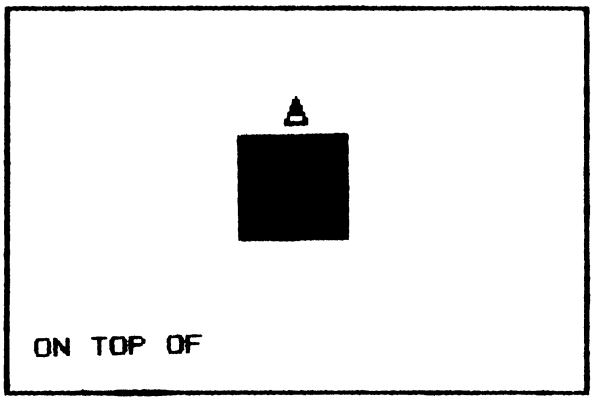
Flower

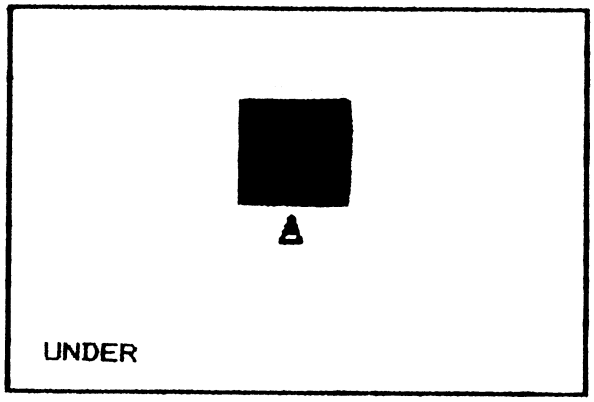
1 half-turn left
Forward 2
Back 2
2 half-turns right
Forward 2
Back 2
1 half-turn left
Forward 6
1 half-turn left
Forward 3
2 half-turns right
Forward 1
2 half turns right
Forward 1
2 half-turns left
Forward 1
2 half-turns right
Forward 1
2 half-turns left
Forward 1
2 half-turns right
Forward 1
2 half-turns right
Forward 3

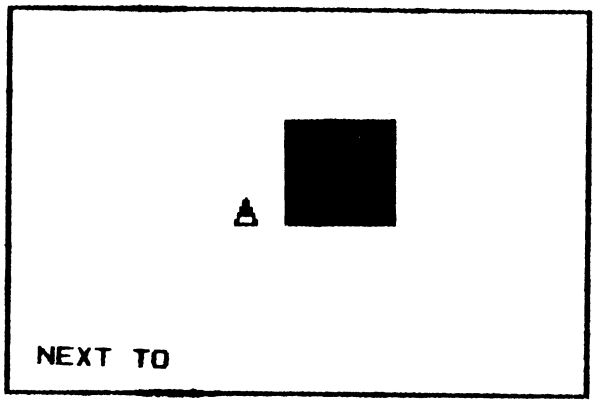


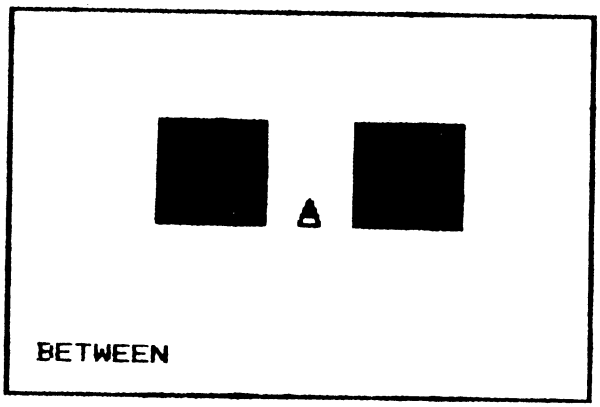

BK
RT
LT
FD

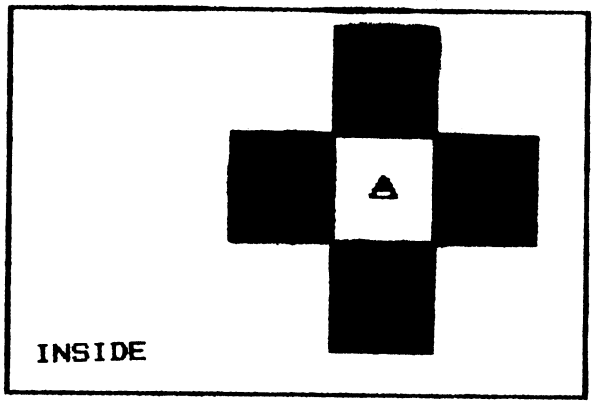
Spatial Relation Cards

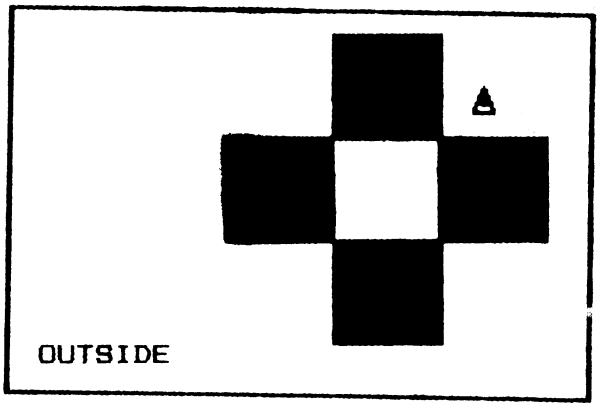




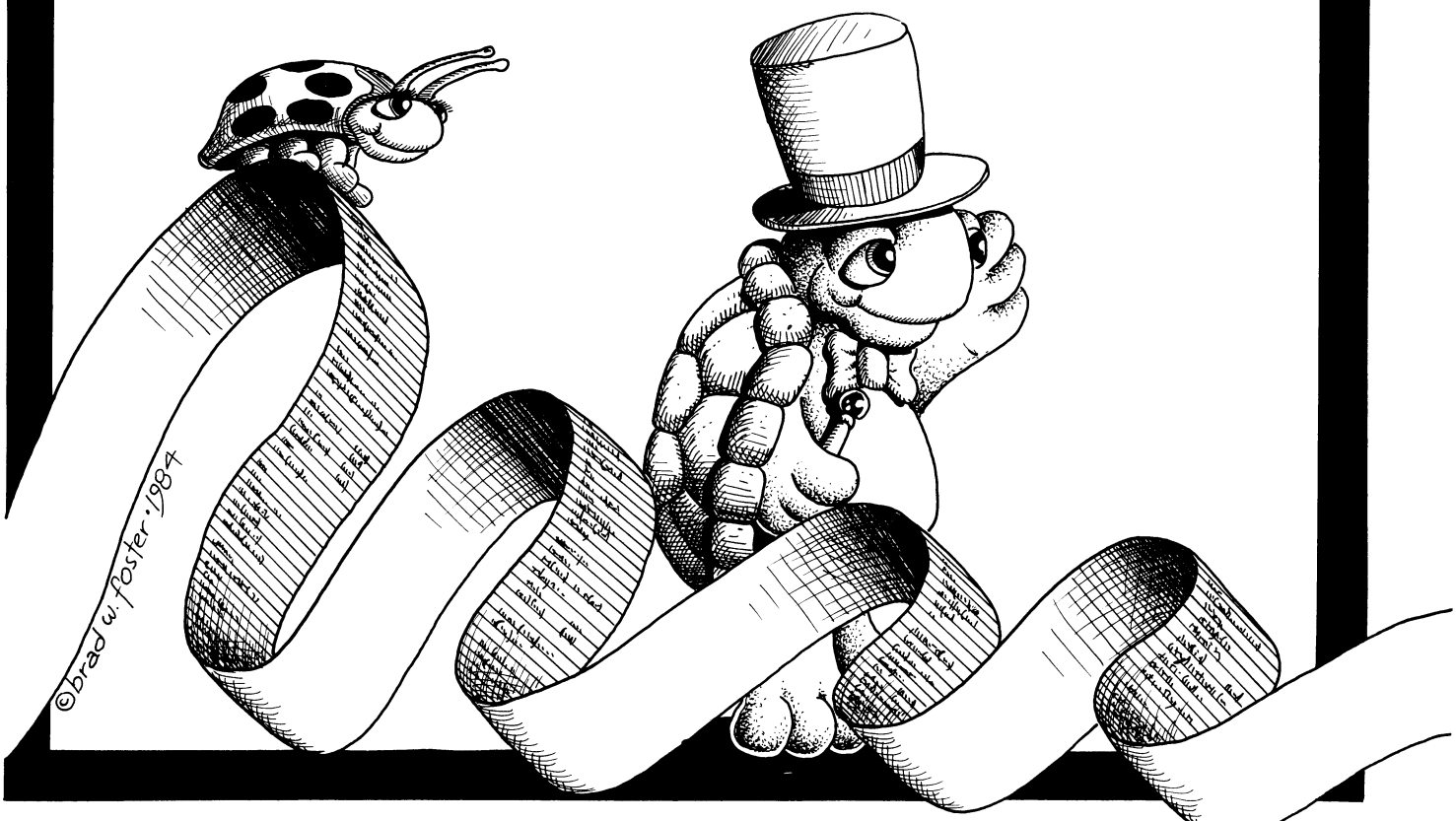


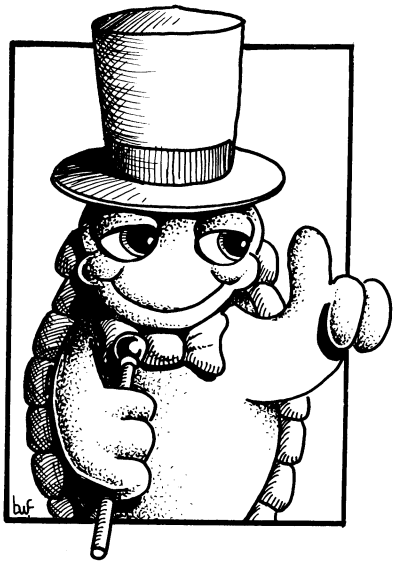






Appendix B
PROCEDURES





To Order a Disk

All of the procedures used in Primarily Logo are included here. You may make your own procedures disk, or for your convenience, you may order a pre-programmed disk. Disks are available for the following versions of Logo: Apple; Terrepin and Krell for the Apple; Commodore 64; Atari; and IBM and Dr. Logo for the IBM.

To order a procedures disk, send a check for $9.95 (add $1 for out of country) to:

Martin-Bearden, Inc.
P.O. Box 337
Grapevine, Texas 76051

REMEMBER to specify which version of Logo you are using!!!

Part I Procedures

(LCSI Logo)

Draw.It.Level1

```
TO PIC.1
INTRO
PRINT [TYPE F 2 TIMES] ANOTHER
PRINT [TYPE R 1 TIME] ANOTHER
PRINT [TYPE F 2 TIMES] ANOTHER
PRINT [TYPE R 1 TIME] ANOTHER
PRINT [TYPE F 2 TIMES] ANOTHER
PRINT [TYPE R 1 TIME] ANOTHER
PRINT [TYPE F 4 TIMES] ANOTHER
PRINT [TYPE R 1 TIME] ANOTHER
PRINT [TYPE F 2 TIMES] ANOTHER
PRINT [TYPE R 1 TIME] ANOTHER
PRINT [TYPE F 3 TIMES] ANOTHER
PRINT [TYPE L 1 TIME] ANOTHER
PRINT [TYPE F 2 TIMES] ANOTHER
PRINT [TYPE L 1 TIME] ANOTHER
PRINT [TYPE F 2 TIMES] ANOTHER
PRINT [TYPE L 1 TIME] ANOTHER
PRINT [TYPE F 2 TIMES] ANOTHER
HT
PRINT [DO YOU SEE SOME BLOCKS?]
END

TO INTRO
CS CLEARTEXT TEXTSCREEN
PRINT [TYPE THE COMMANDS AND]
PRINT [THEN TYPE N FOR NEXT COMMAND.]
WAIT 200
ST
END

TO COMMAND
MAKE "COM READCHAR
IF  :COM = "F [FD 20]
IF  :COM = "R [RT 90]
IF  :COM = "L [LT 90]
IF  :COM = "B [BK 20]
IF  :COM = "N [STOP]
IF  :COM = "E [PENERASE]
IF  :COM = "D [PENDOWN]
COMMAND
END

TO ANOTHER
COMMAND
PRINT [ ]
PRINT [ ]
END
```

```
TO PIC.2
INTRO
REPEAT 4 [SIDE]
HT
PRINT [DO YOU SEE A CROSS?]
END

TO SIDE
PRINT [TYPE L 1 TIME] ANOTHER
PRINT [TYPE F 2 TIMES] ANOTHER
PRINT [TYPE R 1 TIME] ANOTHER
PRINT [TYPE F 2 TIMES] ANOTHER
PRINT [TYPE R 1 TIME] ANOTHER
PRINT [TYPE F 2 TIMES] ANOTHER
END

TO PIC.3
INTRO
PRINT [TYPE R 1 TIME] ANOTHER
PRINT [TYPE F 2 TIMES] ANOTHER
PRINT [TYPE R 1 TIME] ANOTHER
PRINT [TYPE F 3 TIMES] ANOTHER
PRINT [TYPE L 1 TIME] ANOTHER
PRINT [TYPE F 4 TIMES] ANOTHER
PRINT [TYPE L 1 TIME] ANOTHER
PRINT [TYPE F 5 TIMES] ANOTHER
REPEAT 4 [ROOF]
PRINT [TYPE L 1 TIME] ANOTHER
PRINT [TYPE F 2 TIMES] ANOTHER
REPEAT 4 [ROOF]
PRINT [TYPE L 1 TIME] ANOTHER
PRINT [TYPE F 5 TIMES] ANOTHER
PRINT [TYPE L 1 TIME] ANOTHER
PRINT [TYPE F 4 TIMES] ANOTHER
PRINT [TYPE L 1 TIME] ANOTHER
PRINT [TYPE F 3 TIMES] ANOTHER
HT
PRINT [DO YOU SEE A HOUSE?]
END

TO ROOF
PRINT [TYPE L 1 TIME] ANOTHER
PRINT [TYPE F 1 TIME] ANOTHER
PRINT [TYPE R 1 TIME] ANOTHER
PRINT [TYPE F 1 TIME] ANOTHER
END
```

Draw.It.Level2

```
TO PIC.1
INTRO
PRINT [TYPE F 3 TIMES] ANOTHER
PRINT [TYPE L 2 TIMES] ANOTHER
PRINT [TYPE F 4 TIMES] ANOTHER
PRINT [TYPE R 3 TIMES] ANOTHER
PRINT [TYPE F 8 TIMES] ANOTHER
PRINT [TYPE R 2 TIMES] ANOTHER
PRINT [TYPE F 8 TIMES] ANOTHER
PRINT [TYPE R 3 TIMES] ANOTHER
PRINT [TYPE F 4 TIMES] ANOTHER
PRINT [TYPE L 2 TIMES] ANOTHER
PRINT [TYPE F 3 TIMES] ANOTHER
PRINT [TYPE R 2 TIMES] ANOTHER
PRINT [TYPE F 3 TIMES] ANOTHER
HT
PRINT [DO YOU SEE AN ARROW?]
END

TO INTRO
CS CLEARTEXT TEXTSCREEN
PRINT [TYPE THE COMMANDS AND]
PRINT [THEN TYPE N FOR NEXT COMMAND.]
WAIT 200 ST
END

TO COMMAND
MAKE "COM READCHAR
IF  :COM = "F [FD 10]
IF  :COM = "R [RT 45]
IF  :COM = "L [LT 45]
IF  :COM = "B [BK 10]
IF  :COM = "N [STOP]
COMMAND
END

TO ANOTHER
COMMAND
PRINT [ ]
PRINT [ ]
END

TO PIC.2
INTRO
PPRINT [TYPE L 2 TIMES] ANOTHER
REPEAT 8 [SIDE.TURN]
POLE
HT PRINT [DO YOU SEE A STOP SIGN?]
END
```

```
TO SIDE.TURN
PRINT [TYPE R 1 TIME] ANOTHER
PRINT [TYPE F 4 TIMES] ANOTHER
END

TO POLE
PRINT [TYPE B 1 TIME] ANOTHER
PRINT [TYPE L 2 TIMES] ANOTHER
PRINT [TYPE F 6 TIMES] ANOTHER
PRINT [TYPE L 2 TIMES] ANOTHER
PRINT [TYPE F 2 TIMES] ANOTHER
PRINT [TYPE L 2 TIMES] ANOTHER
PRINT [TYPE F 6 TIMES] ANOTHER
END

TO PIC.3
INTRO
PRINT [TYPE F 5 TIMES] ANOTHER
PRINT [TYPE R 2 TIMES] ANOTHER
PRINT [TYPE F 2 TIMES] ANOTHER
PRINT [TYPE R 1 TIME] ANOTHER
REPEAT 3 [PRINT [TYPE F 1 TIME] ANOTHER PRINT [TYPE R 1 TIME] ANOTHER]
PRINT [TYPE F 2 TIMES] ANOTHER
PRINT [TYPE B 2 TIMES] ANOTHER
PRINT [TYPE L 3 TIMES] ANOTHER
REPEAT 3 [PRINT [TYPE F 1 TIME] ANOTHER PRINT [TYPE R 1 TIME] ANOTHER]
PRINT [TYPE F 2 TIMES] ANOTHER
HT PRINT [DO YOU SEE THE LETTER "B"?]
END
```

Draw.It.Level3

```
TO INTRO
CS CLEARTEXT TEXTSCREEN
PRINT [TYPE THE COMMANDS AND]
PRINT [THEN TYPE N FOR NEXT COMMAND.]
WAIT 200
ST
END

TO COMMAND
MAKE "COM READCHAR
IF  :COM = "F [FD 10]
IF  :COM = "R [RT 15]
IF  :COM = "L [LT 15]
IF  :COM = "B [BK 10]
IF  :COM = "N [STOP]
IF  :COM = "E [PENERASE]
IF  :COM = "D [PENDOWN]
COMMAND
END
```

```
TO PIC.1
INTRO
PRINT [TYPE F 2 TIMES] ANOTHER
PRINT [TYPE R 2 TIMES] ANOTHER
PRINT [TYPE F 1 TIME] ANOTHER
PRINT [TYPE R 4 TIMES] ANOTHER
PRINT [TYPE F 3 TIMES] ANOTHER
PRINT [TYPE L 5 TIMES] ANOTHER
PRINT [TYPE F 3 TIMES] ANOTHER
PRINT [TYPE R 5 TIMES] ANOTHER
PRINT [TYPE F 5 TIMES] ANOTHER
PRINT [TYPE R 4 TIMES] ANOTHER
PRINT [TYPE F 4 TIMES] ANOTHER
PRINT [TYPE R 2 TIMES] ANOTHER
PRINT [TYPE F 2 TIMES] ANOTHER
PRINT [TYPE R 6 TIMES] ANOTHER
PRINT [TYPE F 2 TIMES] ANOTHER
WHEEL
PRINT [TYPE F 4 TIMES] ANOTHER
WHEEL
PRINT [TYPE F 2 TIMES] ANOTHER
HT PRINT [DO YOU SEE A CAR?]
END

TO WHEEL
PRINT [TYPE L 5 TIMES] ANOTHER
PRINT [TYPE F 1 TIME] ANOTHER
PRINT [TYPE R 5 TIMES] ANOTHER
PRINT [TYPE F 1 TIME] ANOTHER
PRINT [TYPE R 5 TIMES] ANOTHER
PRINT [TYPE F 1 TIME] ANOTHER
PRINT [TYPE L 5 TIMES] ANOTHER
END

TO ANOTHER
COMMAND
PRINT [ ]
PRINT [ ]
END

TO PIC.2
INTRO
REPEAT 24 [CURVE]
HT
PRINT [DO YOU SEE A CIRCLE?]
END

TO CURVE
PRINT [TYPE F 1 TIME] ANOTHER
PRINT [TYPE R 1 TIME] ANOTHER
END
```

```
TO PIC.3
INTRO
PRINT [TYPE R 8 TIMES] ANOTHER
REPEAT 8 [CURVE]
PRINT [TYPE F 1 TIME] ANOTHER
PRINT [TYPE R 2 TIMES] ANOTHER
PRINT [TYPE F 1 TIME] ANOTHER
PRINT [TYPE R 2 TIMES] ANOTHER
REPEAT 9 [CURVE]
PRINT [TYPE L 6 TIMES] ANOTHER
PRINT [TYPE F 4 TIMES] ANOTHER
PRINT [TYPE R 7 TIMES] ANOTHER
PRINT [TYPE F 3 TIMES] ANOTHER
PRINT [TYPE R 7 TIMES] ANOTHER
PRINT [TYPE F 4 TIMES] ANOTHER
HT
PRINT [DO YOU SEE A VASE?]
END
```

Part I Procedures

(MIT Logo)

Draw.It.Level1

```
TO PIC.1
INTRO
PRINT [TYPE F 2 TIMES.] ANOTHER
PRINT [TYPE R 1 TIME.] ANOTHER
PRINT [TYPE F 2 TIMES.] ANOTHER
PRINT [TYPE R 1 TIME.] ANOTHER
PRINT [TYPE F 2 TIMES.] ANOTHER
PRINT [TYPE R 1 TIME.] ANOTHER
PRINT [TYPE F 4 TIMES.] ANOTHER
PRINT [TYPE R 1 TIME.] ANOTHER
PRINT [TYPE F 2 TIMES.] ANOTHER
PRINT [TYPE R 1 TIME.] ANOTHER
PRINT [TYPE F 3 TIMES.] ANOTHER
PRINT [TYPE L 1 TIME.] ANOTHER
PRINT [TYPE F 2 TIMES.] ANOTHER
PRINT [TYPE L 1 TIME.] ANOTHER
PRINT [TYPE F 2 TIMES.] ANOTHER
PRINT [TYPE L 1 TIME.] ANOTHER
PRINT [TYPE F 2 TIMES.] ANOTHER
HT
PRINT [DO YOU SEE SOME BLOCKS?]
END

TO INTRO
DRAW CLEARTEXT
PRINT [TYPE THE COMMANDS AND]
PRINT [THEN TYPE N FOR NEXT COMMAND.]
WAIT 200
ST
END
```

```
TO WAIT  :TIME
IF  :TIME = 0 STOP
WAIT  :TIME - 1
END

TO COMMAND
MAKE "COM READCHARACTER
IF  :COM = "F FD 20
IF  :COM = "R RT 90
IF  :COM = "L LT 90
IF  :COM = "B BK 20
IF  :COM = "N STOP
IF  :COM = "E PENERASE
IF  :COM = "D PENDOWN
COMMAND
END

TO ANOTHER
COMMAND
PRINT [ ]
PRINT [']
END

TO PIC.2
INTRO
REPEAT 4 [SIDE]
HT
PRINT [DO YOU SEE A CROSS?]
END

TO SIDE
PRINT [TYPE L 1 TIME.] ANOTHER
PRINT [TYPE F 2 TIMES.] ANOTHER
PRINT [TYPE R 1 TIME.] ANOTHER
PRINT [TYPE F 2 TIMES.] ANOTHER
PRINT [TYPE R 1 TIME.] ANOTHER
PRINT [TYPE F 2 TIMES.] ANOTHER
END

TO ROOF
PRINT [TYPE L 1 TIME.] ANOTHER
PRINT [TYPE F 1 TIME.] ANOTHER
PRINT [TYPE R 1 TIME.] ANOTHER
PRINT [TYPE F 1 TIME.] ANOTHER
END
```

```
TO PIC.3
INTRO
PRINT [TYPE R 1 TIME.] ANOTHER
PRINT [TYPE F 2 TIMES.] ANOTHER
PRINT [TYPE R 1 TIME.] ANOTHER
PRINT [TYPE F 3 TIMES.] ANOTHER
PRINT [TYPE L 1 TIME.] ANOTHER
PRINT [TYPE F 4 TIMES.] ANOTHER
PRINT [TYPE L 1 TIME.] ANOTHER
PRINT [TYPE F 5 TIMES.] ANOTHER
REPET 4 [ROOF]
PRINT [TYPE L 1 TIME.] ANOTHER
PRINT [TYPE F 2 TIMES.] ANOTHER
REPEAT 4 [ROOF]
PRINT [TYPE L 1 TIME.] ANOTHER
PRINT [TYPE F 5 TIMES.] ANOTHER
PRINT [TYPE L 1 TIME.] ANOTHER
PRINT [TYPE F 4 TIMES.] ANOTHER
PRINT [TYPE L 1 TIME.] ANOTHER
PRINT [TYPE F 3 TIMES.] ANOTHER
HT
PRINT [DO YOU SEE A HOUSE?]
END
```

Draw.It.Level2

```
TO PIC.1
INTRO
PRINT [TYPE F 3 TIMES.] ANOTHER
PRINT [TYPE L 2 TIMES.] ANOTHER
PRINT [TYPE F 4 TIMES.] ANOTHER
PRINT [TYPE R 3 TIMES.] ANOTHER
PRINT [TYPE F 8 TIMES.] ANOTHER
PRINT [TYPE R 2 TIMES.] ANOTHER
PRINT [TYPE F 8 TIMES.] ANOTHER
PRINT [TYPE R 3 TIMES.] ANOTHER
PRINT [TYPE F 4 TIMES.] ANOTHER
PRINT [TYPE L 2 TIMES.] ANOTHER
PRINT [TYPE F 3 TIMES.] ANOTHER
PRINT [TYPE R 2 TIMES.] ANOTHER
PRINT [TYPE F 3 TIMES.] ANOTHER
HT
PRINT [DO YOU SEE AN ARROW?]
END

TO INTRO
DRAW CLEARTEXT
PRINT [TYPE THE COMMANDS AND]
PRINT [THEN TYPE N FOR NEXT COMMAND.]
WAIT 200
ST
END
```

```
TO COMMAND
MAKE "COM READCHARACTER
IF  :COM = "F FD 10
IF  :COM = "R RT 45
IF  :COM = "L LT 45
IF  :COM = "B BK 10
IF  :COM = "N STOP
IF  :COM = "E PENERASE
IF  :COM = "D PENDOWN
COMMAND
END

TO WAIT  :TIME
IF  :TIME = 0 STOP
WAIT  :TIME - 1
END

TO ANOTHER
COMMAND
PRINT [ ]
PRINT [ ]
END

TO PIC.2
INTRO'
PRINT [TYPE L 2 TIMES.] ANOTHER
REPEAT 8 [SIDE.TURN]
POLE
HT
PRINT [DO YOU SEE A STOP SIGN?]
END

TO SIDE.TURN
PRINT [TYPE R 1 TIME.] ANOTHER
PRINT [TYPE F 4 TIMES.] ANOTHER
END

TO POLE
PRINT [TYPE B 1 TIME.] ANOTHER
PRINT [TYPE L 2 TIMES.] ANOTHER
PRINT [TYPE F 6 TIMES.] ANOTHER
PRINT [TYPE L 2 TIMES.] ANOTHER
PRINT [TYPE F 2 TIMES.] ANOTHER
PRINT [TYPE L 2 TIMES.] ANOTHER
PRINT [TYPE F 6 TIMES.] ANOTHER
END
```

```
TO PIC.3
INTRO
PRINT [TYPE F 5 TIMES.] ANOTHER
PRINT [TYPE R 2 TIMES.] ANOTHER
PRINT [TYPE F 2 TIMES.] ANOTHER
PRINT [TYPE R 1 TIME.] ANOTHER
REPEAT 3 [PRINT [TYPE F 1 TIME.] ANOTHER PRINT [TYPE R 1 TIME.] ANOTHER]
PRINT [TYPE F 2 TIMES.] ANOTHER
PRINT [TYPE B 2 TIMES.] ANOTHER
PRINT [TYPE L 3 TIMES.] ANOTHER
REPEAT 3 [PRINT [TYPE F 1 TIME.] ANOTHER PRINT [TYPE R 1 TIME.] ANOTHER]
PRINT [TYPE F 2 TIMES.] ANOTHER
HT
PRINT [DO YOU SEE THE LETTER "B"?]
END
```

Draw.It.Level3

```
TO PIC.1
INTRO
PRINT [TYPE F 2 TIMES.] ANOTHER
PRINT [TYPE R 2 TIMES.] ANOTHER
PRINT [TYPE F 1 TIME.] ANOTHER
PRINT [TYPE R 4 TIMES.] ANOTHER
PRINT [TYPE F 3 TIMES.] ANOTHER
PRINT [TYPE L 5 TIMES.] ANOTHER
PRINT [TYPE F 3 TIMES.] ANOTHER
PRINT [TYPE R 5 TIMES.] ANOTHER
PRINT [TYPE F 5 TIMES.] ANOTHER
PRINT [TYPE R 4 TIMES.] ANOTHER
PRINT [TYPE F 4 TIMES.] ANOTHER
PRINT [TYPE R 2 TIMES.] ANOTHER
PRINT [TYPE F 2 TIMES.] ANOTHER
PRINT [TYPE R 6 TIMES.] ANOTHER
PRINT [TYPE F 2 TIMES.] ANOTHER
WHEEL
PRINT [TYPE F 4 TIMES.] ANOTHER
WHEEL
PRINT [TYPE F 2 TIMES.] ANOTHER
HT PRINT [DO YOU SEE A CAR?]
END

TO WHEEL
PRINT [TYPE L 5 TIMES.] ANOTHER
PRINT [TYPE F 1 TIME.] ANOTHER
PRINT [TYPE R 5 TIMES.] ANOTHER
PRINT [TYPE F 1 TIME.] ANOTHER
PRINT [TYPE R 5 TIMES.] ANOTHER
PRINT [TYPE F 1 TIME.] ANOTHER
PRINT [TYPE L 5 TIMES.] ANOTHER
END
```

```
TO INTRO
DRAW CLEARTEXT
PRINT [TYPE THE COMMANDS AND]
PRINT [THEN TYPE N FOR NEXT COMMAND.]
WAIT 200 ST
END

TO WAIT  :TIME
IF  :TIME = 0 STOP
WAIT  :TIME - 1
END

TO COMMAND
MAKE "COM READCHARACTER
IF  :COM = "F FD 10
IF  :COM = "R RT 15
IF  :COM = "L LT 15
IF  :COM = "B BK 10
IF  :COM = "N STOP
IF  :COM = "E PENERASE
IF  :COM = "D PENDOWN
COMMAND
END

TO ANOTHER
COMMAND
PRINT [ ]
PRINT [ ]
END

TO PIC.2
INTRO
REPEAT 24 [CURVE] HT
PRINT [DO YOU SEE A CIRCLE?]
END

TO CURVE
PRINT [TYPE F 1 TIME.] ANOTHER
PRINT [TYPE R 1 TIME.] ANOTHER
END

TO PIC.3
INTRO
PRINT [TYPE R 8 TIMES.] ANOTHER
REPEAT 8 [CURVE]
REPEAT 2 [PRINT [TYPE F 1 TIME.] ANOTHER PRINT [TYPE R 2 TIMES.] ANOTHER]
REPEAT 9 [CURVE]
PRINT [TYPE L 6 TIMES.] ANOTHER
PRINT [TYPE F 4 TIMES.] ANOTHER
PRINT [TYPE R 7 TIMES.] ANOTHER
PRINT [TYPE F 3 TIMES.] ANOTHER
PRINT [TYPE R 7 TIMES.] ANOTHER
PRINT [TYPE F 4 TIMES.] ANOTHER
PRINT [DO YOU SEE A VASE?]
HT
END
```

Part II

(LCSI and MIT Logo)

```
TO Y
FD 40 BK 30 RT 10
END

TO T
FD 40 BK 30 LT 10
END

TO X
FD 20 BK 20 RT 10
END

TO FUZZY
REPEAT 36 [Y]
END

TO WUZZY
REPEAT 12 [Y]
PU HOME PD
RT 180
REPEAT 12 [Y]
END

TO EYES
REPEAT 17 [Y]
PU HOME PD
REPEAT 17 [T]
PU HOME PD
RT 180
REPEAT 15 [Y]
PU HOME PD
RT 180
REPEAT 15 [T]
END

TO LOOK
EYES
PU HOME RT 90 FD 55 PD
EYEBALL
PU RT 180 FD 110 PD
EYEBALL
END

TO EYEBALL
REPEAT 36 [X]
END

TO W
FD 40 RT 90 FD 10 RT 90 FD 40 RT 180
END

TO COMB
REPEAT 10 [W]
END

TO COMB2
COMB
RT 180
COMB
END

TO COMB3
COMB
PU FD 80 RT 180 PD
COMB
END

TO SPIN
REPEAT 8 [W RT 45]
END

TO SPIN2
REPEAT 12 [W RT 30]
END

TO SPIN3
SPIN2
PU BK 20 RT 90 FD 5 LT 90 PD
REPEAT 36 [X]
END

TO BOX
REPEAT 4 [W]
LT 90
REPEAT 4 [W]
END

TO BOX2
BOX
RT 60
BOX
FD 40 LT 30
BOX
END

TO BOX3
PU BK 70 LT 90 FD 70 RT 90 PD
REPEAT 4 [BOX RT 90]
END
```

```
TO BOX4
HT
REPEAT 36 [BOX RT 90]
END

TO STAR
REPEAT 8 [V]
END

TO STAR2
REPEAT 8 [V FD 20]
END

TO STAR3
REPEAT 8 [V FD 30]
END

TO STAR4
REPEAT 5 [V2]
END

TO STAR5
REPEAT 18 [V3]
END

TO STAR6
REPEAT 18 [V4]
END


TO V
FD 60 RT 150 FD 60 RT 150
END

TO V2
FD 60 RT 160 FD 60 RT 160
END

TO V3
FD 60 RT 130 FD 60 RT 130
END

TO V4
FD 60 RT 170 FD 60 RT 170
END

TO SUN
PU FD 40 LT 90 FD 110 RT 90 PD
STAR5
END
```

```
TO SCENE
HT
SUN
PICKET
PU HOME FD 30 PD
FLOWER
PU RT 30 FD 80 SETH 0 PD
FLOWER
PU RT 70 FD 40 SETH 0 PD
FLOWER
PU RT 20 FD 70 SETH 0 PD
FLOWER
PU LT 10 FD 90 SETH 0 PD
FLOWER
END


TO PICKET
PU HOME BK 80 PD
REPEAT 28 [W]
END

TO FLOWER
REPEAT 36 [X]
BK 50
END

TO CIRCLE
REPEAT 360 [FD 1 RT 1]
END

TO CIRCLE2
REPEAT 180 [FD 1 RT 2]
END

TO CIRCLE3
REPEAT 90 [FD 1 RT 4]
END

TO CIRCLE4
REPEAT 240 [FD 1 RT 1.5]
END

TO CIRCLE5
REPEAT 120 [FD 1 RT 3]
END

TO CIRCLES
REPEAT 10 [CIRCLE2 FD 10]
END
```

```
TO CIRCLES2
REPEAT 3 [CIRCLE2 RT 120]
END

TO CIRCLES3
PU LT 90 FD 100 RT 90 PD
CIRCLE
PU RT 90 FD 115 LT 90 PD
CIRCLE2
PU RT 90 FD 58 LT 90 PD
CIRCLE3
END

TO CIRCLES4
CIRCLE
CIRCLE2
CIRCLE3
CIRCLE4
CIRCLE5
END

TO SQUARES
SQUARE3
PU FD 10 RT 90 FD 10 LT 90 PD
SQUARE2
PU FD 10 RT 90 FD 10 LT 90 PD
SQUARE
END

TO SQUARES2
SQUARE3
FD 10 LT 90
SQUARE2
FD 40 RT 90 FD 10 LT 90
SQUARE
PU HOME PD
RT 90 FD 60 LT 90 FD 50 RT 90
SQUARE2
FD 40 RT 90 FD 10 LT 90
SQUARE
END

TO SQUARES3
REPEAT 6 [SQUARE RT 90 FD 20 LT 30]
END

TO SQUARES4
REPEAT 8 [SQUARE2 RT 45]
END

TO SQUARES5
REPEAT 12 [RT 30 SQUARE FD 20]
END

TO B
FD 60 RT 150 FD 60 RT 150 FD 60
END

TO B2
FD 60 RT 100 FD 60 RT 100 FD 60
END

TO TRIANGLE
REPEAT 3 [FD 60 RT 120]
END

TO TRIANGLE2
REPEAT 3 [FD 30 RT 120]
END

TO TRIANGLES
REPEAT 4 [TRIANGLE RT 60 FD 30 RT 30]
END

TO TRIANGLES2
REPEAT 18 [TRIANGLE RT 20]
END

TO TRIANGLES3
REPEAT 6 [TRIANGLE RT 60]
END

TO TRIANGLES4
REPEAT 4 [TRIANGLE RT 90]
END

TO BALANCE
PU BK 50 PD
CIRCLE2
REPEAT 35 [FD 1 RT 2]
LT 180 FD 10 RT 90
SQUARE
FD 20 RT 90 FD 20 RT 30 FD 8 BK 16 LT 90
REPEAT 2 [FD 50 RT 90 FD 16 RT 90]
FD 50 LT 120 FD 10 RT 120
TRIANGLE2
END
```

Part III

(LCSI Logo)

```
?POPS
TO CHAT
PRINT [WHAT IS YOUR NAME?]
PRINT SENTENCE [HELLO,] READLIST
PRINT [WHAT IS YOUR FAVORITE SPORT?]
PRINT SENTENCE [I DON'T LIKE] READLIST
PRINT [WHO'S YOUR BEST FRIEND?]
PRINT SENTENCE READLIST [IS SILLY!]
END

TO SILLY.SENTENCE
PRINT [NAME A NOUN.]
MAKE "NOUN READLIST
PRINT [NAME A VERB.]
MAKE "VERB READLIST
PRINT [NAME AN ADJECTIVE.]
MAKE "ADJECTIVE READLIST
PRINT [NAME AN ADVERB]
MAKE "ADVERB READLIST
CLEARTEXT
PRINT (SENTENCE  :ADJECTIVE  :NOUN  :VERB  :ADVERB)
END

TO STORY
PRINT [NAME A BOY.]
MAKE "BOY READLIST
PRINT [NAME A GIRL.]
MAKE "GIRL READLIST
PRINT [NAME SOMETHING TO CLIMB ON.]
MAKE "CLIMB READLIST
PRINT [NAME A CONTAINER.]
MAKE "CONTAINER READLIST
PRINT [NAME SOMETHING TO DRINK.]
MAKE "DRINK READLIST
PRINT [NAME A PART OF THE BODY.]
MAKE "BODY READLIST
PRINT [NAME A VERB ENDING WITH ING.]
MAKE "VERB READLIST
CLEARTEXT
PRINT (SENTENCE  :BOY [AND]  :GIRL)
PRINT SENTENCE [TO FETCH A]  :CONTAINER [OF]  :DRINK)
PRINT SENTENCE:BOY [FELL DOWN.]
PRINT SENTENCE [AND BROKE HIS]  :BODY
PRINT (SENTENCE [AND]  :GIRL [CAME]  :VERB [AFTER.])
END
```

```
TO QUESTION
PRINT [WHAT IS THE CAPITAL OF TEXAS?]
TEST READLIST = [AUSTIN]
IFTRUE [PRINT [THAT'S CORRECT!]]
IFFALSE [PRINT [NO, THE CAPITAL OF TEXAS IS AUSTIN.]]
END

TO GEOGRAPHY
PRINT [IF I WAS GOING FROM TEXAS TO CALIFORNIA,]
PRINT [WHAT IS THE FIRST STATE I WOULD COME TO?]
TEST READLIST = [NEW MEXICO]
IFTRUE [PRINT [RIGHT!]]
IFFALSE [PRINT [GO LOOK AT A MAP!] GEOGRAPHY]
PRINT [WHAT FAMOUS CAVE IS IN NEW MESICO?]
TEST READLIST = [CARLSBAD CAVERNS]
IFTRUE [PRINT [YES!]]
IFFALSE [PRINT [NO, IT'S CARLSBAD CAVERNS.]]
END

TO QUIZ
QUESTION1
QUESTION2
END

TO QUESTION1
CLEARTEXT TEXTSCREEN
PRINT [WHAT IS A STALACTITE?]
PRINT [TYPE A, B, OR C.]
PRINT [ ]
PRINT [A. A FORMATION THAT HANGS FROM THE ROOF OF A CAVE.]
PRINT [B. A PAINFUL CAVITY.]
PRINT [C. A MALE DEER WITH A FULL HEAD OF ANTLERS.]
TEST READLIST = [A]
IFTRUE [PRINT [RIGHT!] WAIT 50]
IFFALSE [PRINT [NO, LET ME SHOW YOU.] WAIT 50]
STALACTITE
END

TO STALACTITE
CS HT
PRINT [A STALACTITE IS A CRYSTAL FORMATION THAT]
PRINT [HANGS FROM THE ROOF OF A CAVE.]
CAVE
PRINT [A STALAGMITE IS A CRYSTAL FORMATION THAT]
PRINT [GROWS UP FROM THE FLOOR OF THE CAVE.]
CAVE2
WAIT 100 CS
END
```

```
TO CAVE SETPOS [- 70 0] SETH 0 PD
REPEAT 19 [FD 10 RT 10]
RT 80 FD 11 5
PU SETPOS [- 70 0] SETH 0 PD
REPEAT 6 [FD 10 RT 10]
REPEAT 8 [CRYSTAL FD 10 RT 10]
REPEAT 5 [FD 10 RT 10]
WAIT 50
END

TO CRYSTAL
RT 90 FD 15 BK 15 LT 90
END

TO CAVE2
RT 80
REPEAT 8 [FD 12 TRI]
END

TO TRI
REPEAT 3 [FD 7 RT 120]
END

TO QUESTION2
CLEARTEXT TEXTSCREEN
PRINT [WHAT IS A PERSON CALLED WHO STUDIES CAVES?]
PRINT [TYPE A, B, OR C.]
PRINT [ ]
PRINT [A. A CAVEMAN]
PRINT [B. A SPELUNKER]
PRINT [C. AN AVIATOR]
TEST READLIST = [B]
IFTRUE [PRINT [VERY GOOD!]]
IFFALSE [PRINT [TRY AGAIN!] WAIT 50 QUESTION2]
END
```

Part III

(MIT Logo)

```
TO CHAT
PRINT [WHAT IS YOUR NAME?]
PRINT SENTENCE [HELLO,] REQUEST
PRINT [WHAT IS YOUR FAVORITE SPORT?]
PRINT SENTENCE [I DON'T LIKE] REQUEST
PRINT [WHO'S YOUR BEST FRIEND?]
PRINT SENTENCE REQUEST [IS SILLY!]
END

TO SILLY.SENTENCE
PRINT [NAME AN ADJECTIVE]
MAKE "ADJECTIVE REQUEST
PRINT [NAME A NOUN]
MAKE "NOUN REQUEST
PRINT [NAME A VERB]
MAKE "VERB REQUEST
PRINT [NAME AN ADVERB]
MAKE "ADVERB REQUEST
CLEARTEXT
PRINT ( SENTENCE  :ADJECTIVE  :NOUN  :VERB  :ADVERB )
END

TO STORY
PRINT [NAME A BOY.]
MAKE "BOY REQUEST
PRINT [NAME A GIRL.]
MAKE "GIRL REQUEST
PRINT [NAME SOMETHING TO CLIMB ON.]
MAKE "CLIMB REQUEST
PRINT [NAME A CONTAINER]
MAKE "CONTAINER REQUEST
PRINT [NAME SOMETHING TO DRINK.]
MAKE "DRINK REQUEST
PRINT [NAME A PART OF THE BODY.]
MAKE "BODY REQUEST
PRINT [NAME A VERB ENDING WITH ING.]
MAKE "VERB REQUEST
CLEARTEXT
PRINT ( SENTENCE  :BOY [AND]  :GIRL )
PRINT SENTENCE [WENT UP THE]  :CLIMB
PRINT ( SENTENCE [TO FETCH A]  :CONTAINER [OF]  :DRINK )
PRINT SENTENCE  :BOY [FELL DOWN]
PRINT SENTENCE [AND BROKE HIS]  :BODY
PRINT ( SENTENCE [AND]  :GIRL [CAME]  :VERB [AFTER.] )
END
```

```
TO QUESTION
PRINT [WHAT IS THE CAPITAL OF TEXAS?]
TEST REQUEST = [AUSTIN]
IFTRUE PRINT [THAT'S CORRECT!]
IFFALSE PRINT [NO, THE CAPITAL OF TEXAS IS AUSTIN.]
END

TO GEOGRAPHY
PRINT [IF I WAS GOING FROM TEXAS TO CALIFORNIA,]
PRINT [WHAT IS THE FIRST STATE I WOULD COME TO?]
TEST REQUEST = [NEW MEXICO]
IFTRUE PRINT [RIGHT!]
IFFALSE PRINT [GO LOOK AT A MAP!] GEOGRAPHY
PRINT [WHAT FAMOUS CAVE IS IN NEW MEXICO?]
TEST REQUEST = [CARLSBAD CAVERNS]
IFTRUE PRINT [YES!]
IFFALSE PRINT [NO, IT'S CARLSBAD CAVERNS.]
END

TO QUIZ
QUESTION1
QUESTION2
END

TO QUESTION1
CLEARTEXT TEXTSCREEN
PRINT [WHAT IS A STALACTITE?]
PRINT [TYPE A, B, OR C.]
PRINT [ ]
PRINT [A. A FORMATION THAT HANGS FROM THE ROOF OF A CAVE.]
PRINT [B. A PAINFUL CAVITY.]
PRINT [C. A MALE DEER WITH A FULL HEAD OF ANTLERS.]
TEST REQUEST = [A]
IFTRUE PRINT [RIGHT] WAIT 50
IFFALSE PRINT [NO, LET ME SHOW YOU.] WAIT 50
STALACTITE
END

TO WAIT  :TIME
IF  :TIME = 0 STOP
WAIT  :TIME - 1
END

TO STALACTITE
CS HT
PRINT [A STALACTITE IS A CRYSTAL FORMATION THAT]
PRINT [HANGS FROM THE ROOF OF A CAVE.]
CAVE
PRINT [A STALAGMITE IS A CRYSTAL FORMATION THAT]
PRINT [GROWS UP FROM THE FLOOR OF THE CAVE.]
CAVE2
WAIT 100 CS
END
```

```
TO CAVE
PU SETXY - 70 0 SETH 0 PD
REPEAT 19 [FD 10 RT 10]
RT 80 FD 115
PU SETXY - 70 0 SETH 0 PD
REPEAT 6 [FD 10 RT 10]
REPEAT 8 [CRYSTAL FD 10 RT 10]
REPEAT 5 [FD 10 RT 10]
WAIT 50
END

TO CRYSTAL
RT 90 FD 15 BK 15 LT 90
END

TO CAVE2
RT 80
REPEAT 8 [FD 12 TRI]
END

TO TRI
REPEAT 3 [FD 7 RT 120]
END

TO QUESTION2
CLEARTEXT TEXTSCREEN
PRINT [WHAT IS A PERSON CALLED WHO STUDIES CAVES?]
PRINT [TYPE A, B, OR C.]
PRINT [ ]
PRINT [A. A CAVEMAN]
PRINT [B. A SPELUNKER]
PRINT [C. AN AVIATOR]
TEST REQUEST = [B]
IFTRUE PRINT [VERY GOOD]
IFFALSE PRINT [TRY AGAIN!] WAIT 100 QUESTION2
END
```

Part IV Procedures

(LCSI Logo)

Getgoing

```
TO GETGOING  :DISTANCE
PU
FD  :DISTANCE
COMMAND
GETGOING  :DISTANCE
END

TO COMMAND
MAKE "COM READKEY
IF  :COM = "NORTH [SETH 0]
IF  :COM = "SOUTH [SETH 180]
IF  :COM = "EAST [SETH 90]
IF  :COM = "WEST [SETH 270]
IF  :COM = "AROUND [CIRCLE]
IF  :COM = "FASTER [MAKE "DISTANCE  :DISTANCE + 1]
IF  :COM = "SLOWER [CHECK]
IF  :COM = "STOP [MAKE "DISTANCE 0]
IF  :COM = "GO [MAKE "DISTANCE 1]
END

TO CIRCLE
REPEAT 120 [FD 1 RT 3]
END

TO CHECK
TEST  :DISTANCE > 0
IFTRUE [MAKE "DISTANCE  :DISTANCE - 1]
IFFALSE [STOP]
END

TO READKEY
IF KEYP [CLEARTEXT OUTPUT READWORD]
OUTPUT ''
END
```

Getgoing.Again

```
TO GETGOING.AGAIN  :DISTANCE
PU
FD  :DISTANCE
COMMAND
GETGOING.AGAIN  :DISTANCE
END

TO READKEY
IF KEYP [CLEARTEXT OUTPUT READWORD]
OUTPUT ''
END
```

```
TO COMMAND
MAKE "COM READKEY
IF  :COM = "NORTH [SETH 0 MAKE "DISTANCE 0 STOP]
IF  :COM = "SOUTH [SETH 180 MAKE "DISTANCE 0 STOP]
IF  :COM = "EAST [SETH 90 MAKE "DISTANCE 0 STOP]
IF  :COM = "WEST [SETH 270 MAKE "DISTANCE 0 STOP]
IF  :COM = "AROUND [CIRCLE MAKE "DISTANCE 0 STOP]
IF  :COM = "STOP [MAKE "DISTANCE 0 STOP]
IF  :COM = "GO [MAKE "DISTANCE 1 STOP]
IF  :COM = "BOX [SQUARE MAKE "DISTANCE 0 STOP]
IF  :COM = "CLEAR [CLEARSCREEN]
END

TO CIRCLE
REPEAT 120 [FD 1 RT 3]
END

TO SQUARE
PD
REPEAT 4 [FD 20 RT 90]
REPEAT 10 [FD 20 RT 90 FD 1 RT 90 FD 20 LT 90 FD 1 LT 90]
PU
END
```

Comparisons

```
TO COMPARISONS
COMPARE
COMPARISONS
END

TO COMPARE
SPLITSCREEN HT
MAKE "COM READWORD
IF  :COM = "LONG [PU SETPOS [-100 20] SETH 90 PD FD 50 PU STOP]
IF  :COM = "LONGER [SETPOS [-100 0] SETH 90 PD FD 100 PU STOP]
IF  :COM = "LONGEST [SETPOS [-100 -20] SETH 90 PD FD 150 PU STOP]
IF  :COM = "LARGE [CIRCLE1 STOP]
IF  :COM = "LARGER [CIRCLE2 STOP]
IF  :COM = "LARGEST [CIRCLE3 STOP]
IF  :COM = "SMALL [BOX1 STOP]
IF  :COM = "SMALLER [BOX2 STOP]
IF  :COM = "SMALLEST [BOX3 STOP]
IF  :COM = "CLEAR [CLEARSCREEN CLEARTEXT STOP]
END

TO CIRCLE1
PU SETPOS [-50 20] SETH 0 PD
REPEAT 120 [FD 1 RT 3]
END

TO CIRCLE2
PU SETPOS [-50 20] SETH 0 PD
REPEAT 180 [FD 1 RT 2]
END
```

```
TO CIRCLE3
SETPOS [-50 20] SETH 0 PD
REPEAT 360 [FD 1 RT 1]
END

TO BOX1
SETPOS [-15 0] SETH 0 PD
REPEAT 4 [FD 30 RT 90]
PU
END

TO BOX2
SETPOS [-10 5] SETH 0 PD
REPEAT 4 [FD 20 RT 90]
PU
END

TO BOX3
SETPOS [-5 10] SETH 0 PD
REPEAT 4 [FD 10 RT 90]
PU
END
```

Colors

```
TO COLORS
SPLITSCREEN HT
MAKE "COM READKEY
IF  :COM = "BLUE [SETPC 5 FILL]
IF  :COM = "GREEN [SETPC 2 FILL]
IF  :COM = "WHITE [SETPC 1 FILL]
IF  :COM = "PURPLE [SETPC 3 FILL]
IF  :COM = "ORANGE [SETPC 4 FILL]
COLORS
END

TO FILL
CLEARSCREEN
REPEAT 4 [FD 20 RT 90]
REPEAT 10 [FD 20 RT 90 FD 1 RT 90 FD 20 LT 90 FD 1 LT 90]
END

TO READKEY
IF KEYP [CLEARTEXT OUTPUT READWORD]
OUTPUT ''
END
```

Counting

```
TO COUNTING
HT CS CLEARTEXT
MAKE "NUMBER 1 + RANDOM 9
PU SETPOS [-130 60]
BOXES  :NUMBER -130
GETANSWER
IF NOT  :DIDITRIGHT [STOP]
PRINT [TRY ANOTHER ONE.] WAIT 90
COUNTING
END

TO BOXES  :NUMBER  :X
REPEAT  :NUMBER [SETX  :X PD BOX PU MAKE "X  :X + 27] SETX -130
END

TO BOX
REPEAT 4 [FD 20 RT 90]
END

TO GETANSWER
CLEARTEXT
PRINT [HOW MANY BOXES?]
MAKE "ANSWER READNUMBER
TEST  :ANSWER = :NUMBER
IFTRUE [PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE]
IFFALSE [CLEARTEXT PRINT [HIT THE "C" ONE TIME FOR EACH BOX.] MAKE
  "KEEPTRACK 0 OPERATIONS CHECK]
END

TO READNUMBER
MAKE "ANSWER READLIST
TEST  :ANSWER = [ ]
IFTRUE [PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER]
TEST NOT NUMBERP FIRST  :ANSWER
IFTRUE [PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER]
IFFALSE [OUTPUT FIRST  :ANSWER]
END

TO OPERATIONS
IF  :KEEPTRACK =  :NUMBER [STOP]
MAKE "OP READKEY
IF  :OP = "C [FILL MAKE "KEEPTRACK  :KEEPTRACK + 1]
OPERATIONS
END

TO READKEY
IFKEYP [OUTPUT READCHAR]
OUTPUT ''
END
```

```
TO FILL
PD REPEAT 10 [FD 20 RT 90 FD 1 RT 90 FD 20 LT 90 FD 1 LT 90]
PU RT 90 FD 7 LT 90
END

TO CHECK
CLEARTEXT PRINT [HOW MANY BOXES WERE FILLED?]
MAKE "ANSWER READNUMBER
TEST  :ANSWER =  :NUMBER
IFTRUE [CLEARTEXT PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE]
IFFALSE [CLEARTEXT PRINT [TALK WITH YOUR TEACHER.] MAKE "DIDITRIGHT
  "FALSE]
END
```

Adding

```
TO ADDING
HT CS CLEARTEXT
MAKE "NUMBER 1 + RANDOM 9
MAKE "NUMBER2 1 + RANDOM 9
PU SETPOS [-130 60]
BOXES  :NUMBER -130
ADD
GETANSWER
IF NOT  :DIDITRIGHT [STOP]
PRINT [TRY ANOTHER ONE.] WAIT 90
ADDING
END

TO ADD
PRINT (SENTENCE [HIT THE "A" KEY]  :NUMBER2 [TIMES.])
MAKE "KEEPTRACK 0
OPERATIONS
END

TO OPERATIONS
IF  :KEEPTRACK =  :NUMBER2 [STOP]
MAKE "OP READKEY
IF  :OP = "A [BOXES2 MAKE "KEEPTRACK  :KEEPTRACK + 1]
OPERATIONS
END

TO BOXES  :NUMBER  :X
REPEAT  :NUMBER [SETX  :X PD BOX PU MAKE "X  :X + 27] SETX -130
END

TO BOXES2
PU SETY 0 PD
BOX
PU RT 90 FD 27 LT 90 PD
END

TO BOX
REPEAT 4 [FD 20 RT 90]
END
```

```
TO GETANSWER
CLEARTEXT
PRINT [HOW MANY BOXES?]
MAKE "ANSWER READNUMBER
TEST  :ANSWER =  :NUMBER +  :NUMBER2
IFTRUE [PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE]
IFFALSE [CHECK]
END

TO CHECK
CLEARTEXT
PRINT [COUNT THE BOXES AGAIN.]
PRINT [TYPE THE NUMBER.]
MAKE "ANSWER READNUMBER
TEST  :ANSWER =  :NUMBER +  :NUMBER2
IFTRUE [CLEARTEXT PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE]
IFFALSE [CLEARTEXT PRINT [TALK WITH YOUR TEACHER.] MAKE "DIDITRIGHT
  "FALSE]
END

TO READKEY
IF KEYP [OUTPUT READCHAR]
OUTPUT ''
END

TO READNUMBER
MAKE "ANSWER READLIST
TEST  :ANSWER = [ ]
IFTRUE [PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER]
TEST NOT NUMBERP FIRST  :ANSWER
IFTRUE [PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER]
IFFALSE [OUTPUT FIRST  :ANSWER]
END
```

Subtracting

```
TO SUBTRACTING
HT CS CLEARTEXT
CHOOSENUMBERS
PU SETPOS [-130 60]
BOXES  :NUMBER -130
SUBTRACT
GETANSWER
IF NOT  :DIDITRIGHT [STOP]
PRINT [TRY ANOTHER ONE.] WAIT 90
SUBTRACTING
END

TO CHOOSENUMBERS
MAKE "NUMBER 1 + RANDOM 9
MAKE "NUMBER2 1 + RANDOM 9
IF  :NUMBER2 > :NUMBER [CHOOSENUMBERS]
END
```

```
TO BOXES :NUMBER :X
REPEAT :NUMBER [SETX :X PD BOX PU MAKE "X :X + 27]
END

TO BOXES2
PU SETY 0 PD
BOX
PU RT 90 FD 27 LT 90 PD
END

TO BOX
REPEAT 4 [FD 20 RT 90]
END

TO SUBTRACT
PRINT (SENTENCE [HIT THE "S" KEY] :NUMBER2 [TIMES.])
MAKE "KEEPTRACK 0
OPERATIONS
END

TO OPERATIONS
IF :KEEPTRACK = :NUMBER2 [STOP]
MAKE "OP READKEY
IF :OP = "S [ERASE.BOXES MAKE "KEEPTRACK :KEEPTRACK + 1]
OPERATIONS
END

TO ERASE.BOXES
PE BOX
PU LT 90 FD 7 FD 20 RT 90
END

TO GETANSWER
CLEARTEXT
PRINT [HOW MANY BOXES?]
MAKE "ANSWER READNUMBER
TEST :ANSWER = :NUMBER - :NUMBER2
IFTRUE [PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE]
IFFALSE [CHECK]
END

TO CHECK
CLEARTEXT PRINT [COUNT THE BOXES AGAIN.]
PRINT [TYPE THE NUMBER.]
MAKE "ANSWER READNUMBER
TEST :ANSWER = :NUMBER - :NUMBER2
IFTRUE [CLEARTEXT PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE]
IFFALSE [CLEARTEXT PRINT [TALK WITH YOUR TEACHER.] MAKE "DIDITRIGHT
  "FALSE]
END
```

```
TO READNUMBER
MAKE "ANSWER READLIST
TEST  :ANSWER = [ ]
IFTRUE [PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER]
TEST NOT NUMBERP FIRST  :ANSWER
IFTRUE [PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER]
IFFALSE [OUTPUT FIRST  :ANSWER]
END

TO READKEY
IF KEYP [OUTPUT READCHAR]
OUTPUT ''
END
```

Part IV Procedures

(MIT Logo)

Getgoing

```
TO GETGOING  :DISTANCE
PU
FD  :DISTANCE
COMMAND
GETGOING  :DISTANCE
END

TO COMMAND
MAKE "COM READKEY
IF  :COM = "NORTH SETH 0
IF  :COM = "SOUTH SETH 180
IF  :COM = "EAST SETH 90
IF  :COM = "WEST SETH 270
IF  :COM = "AROUND CIRCLE
IF  :COM = "FASTER MAKE "DISTANCE  :DISTANCE + 1
IF  :COM = "SLOWER CHECK
IF  :COM = "STOP MAKE "DISTANCE 0
IF  :COM = "GO MAKE "DISTANCE 1
END

TO CIRCLE
REPEAT 120 [FD 1 RT 3]
END

TO CHECK
TEST  :DISTANCE > 0
IFTRUE MAKE "DISTANCE  :DISTANCE - 1
IFFALSE STOP
END

TO READKEY
IF RC? CLEARTEXT OUTPUT FIRST REQUEST
OUTPUT''
END
```

Getgoing.Again

```
TO GETGOING.AGAIN  :DISTANCE
PU
FD  :DISTANCE
COMMAND
GETGOING.AGAIN  :DISTANCE
END

TO COMMAND
MAKE "COM READKEY
IF  :COM = "NORTH SETH 0 MAKE "DISTANCE 0 STOP
IF  :COM = "SOUTH SETH 180 MAKE "DISTANCE 0 STOP
IF  :COM = "EAST SETH 90 MAKE "DISTANCE 0 STOP
IF  :COM = "WEST SETH 270 MAKE "DISTANCE 0 STOP
IF  :COM = "AROUND CIRCLE MAKE "DISTANCE 0 STOP
IF  :COM = "STOP MAKE "DISTANCE 0 STOP
IF  :COM = "GO MAKE "DISTANCE 1 STOP
IF  :COM = "BOX SQUARE MAKE "DISTANCE 0 STOP
IF  :COM = "CLEAR CLEARSCREEN
END

TO READKEY
IF RC? CLEARTEXT OUTPUT FIRST REQUEST
OUTPUT ''
END

TO CIRCLE
REPEAT 120 [FD 1 RT 3]
END

TO SQUARE
PD
REPEAT 4 [FD 20 RT 90]
REPEAT 10 [FD 20 RT 90 FD 1 RT 90 FD 20 LT 90 FD 1 LT 90]
PU
END
```

Comparisons

```
TO COMPARE
SPLITSCREEN HT
MAKE "COM READWORD
IF  :COM = "LONG PU SETXY - 100 20 SETH 90 PD FD 50 PU STOP
IF  :COM = "LONGER SETXY - 100 0 SETH 90 PD FD 100 PU STOP
IF  :COM = "LONGEST SETXY - 100 ( - 20 ) SETH 90 PD FD 150 PU STOP
IF  :COM = "LARGE CIRCLE1 STOP
IF  :COM = "LARGER CIRCLE2 STOP
IF  :COM = "LARGEST CIRCLE3 STOP
IF  :COM = "SMALL BOX1 STOP
IF  :COM = "SMALLER BOX2 STOP
IF  :COM = "SMALLEST BOX3 STOP
IF  :COM = "CLEAR CLEARSCREEN CLEARTEXT STOP
END
```

```
TO COMPARISONS
COMPARE
COMPARISONS
END


TO READWORD
OUTPUT FIRST REQUEST
END

TO CIRCLE1
SETXY - 50 20 SETH 0 PD
REPEAT 120 [FD 1 RT 3]
PU
END

TO CIRCLE2
SETXY - 50 20 SETH 0 PD
REPEAT 180 [FD 1 RT 2]
PU
END

TO CIRCLE3
SETXY - 50 20 SETH 0 PD
REPET 360 [FD 1 RT 1]
PU
END
```

Colors

```
TO COLORS
SPLITSCREEN PU HOME PD HT
MAKE "COM READKEY
IF  :COM = "BLUE PC 5 FILL
IF  :COM = "GREEN PC 2 FILL
IF  :COM = "WHITE PC 1 FILL
IF  :COM = "PURPLE PC 3 FILL
IF  :COM = "ORANGE PC 4 FILL
COLORS
END

TO FILL
CLEARSCREEN
REPEAT 4 [FD 20 RT 90]
REPEAT 10 [FD 20 RT 90 FD 1 RT 90 FD 20 LT 90 FD 1 LT 90]
END

TO READKEY
IF RC? CLEARTEXT OUTPUT FIRST REQUEST
OUTPUT ''
END
```

Counting

```
TO COUNTING
HT CS CLEARTEXT
MAKE "NUMBER 1 + RANDOM 9
PU SETXY - 130 60
BOXES :NUMBER ( - 130 )
GETANSWER
IF NOT :DIDITRIGHT STOP
PRINT [TRY ANOTHER ONE.] WAIT 90
COUNTING
END

TO BOXES :NUMBER :X
REPEAT :NUMBER [SETX :X PD BOX PU MAKE "X :X + 27] SETX - 130
END

TO BOX
REPEAT 4 [FD 20 RT 90]
END

TO GETANSWER
CLEARTEXT
PRINT [HOW MANY BOXES?]
MAKE "ANSWER READNUMBER
TEST :ANSWER = :NUMBER
IFTRUE PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE
IFFALSE CLEARTEXT PRINT [HIT THE "C" ONE TIME FOR EACH BOX.] MAKE
  "KEEPTRACK 0
OPERATIONS CHECK
END

TO READNUMBER
MAKE "ANSWER REQUEST
REST :ANSWER = [ ]
IFTRUE PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER
TEST NOT NUMBER? FIRST :ANSWER
IFTRUE PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER
IFFALSE OUTPUT FIRST :ANSWER
END

TO OPERATIONS
IF :KEEPTRACK = :NUMBER STOP
MAKE "OP READKEY
IF :OP = "C FILL MAKE "KEEPTRACK :KEEPTRACK + 1
OPERATIONS
END

TO READKEY
IF RC? OUTPUT READCHARACTER
OUTPUT ''
END
```

```
TO FILL
PD REPEAT 10 [FD 20 RT 90 FD 1 RT 90 FD 20 LT 90 FD 1 LT 90]
PU RT 90 FD 7 LT 90
END

TO CHECK
CLEARTEXT PRINT [HOW MANY BOXES WERE FILLED?]
MAKE "ANSWER READNUMBER
TEST  :ANSWER =  :NUMBER
IFTRUE CLEARTEXT PRINT [GOOD![ MAKE "DIDITRIGHT "TRUE
IFFALSE CLEARTEXT PRINT [TALK WITH YOUR TEACHER.] MAKE "DIDITRIGHT
  "FALSE
END

TO WAIT  :TIME
IF  :TIME = 0 STOP
WAIT  :TIME - 1
END
```

Adding

```
TO ADDING
HT CS CLEARTEXT
MAKE "NUMBER 1 + RANDOM 9
MAKE "NUMBER2 1 + RANDOM 9
PU SETXY - 130 60
BOXES  :NUMBER ( - 130 )
ADD
GETANSWER
IF NOT  :DIDITRIGHT STOP
PRINT [TRY ANOTHER ONE.] WAIT 90
ADDING
END

TO ADD
PRINT ( SENTENCE [HIT THE "A" KEY]  :NUMBER2 [TIMES.])
MAKE "KEEPTRACK 0
OPERATIONS
END

TO OPERATIONS
IF  :KEEPTRACK =  :NUMBER2 STOP
MAKE "OP READKEY
IF  :OP = "A BOXES2 MAKE "KEEPTRACK  :KEEPTRACK + 1
OPERATIONS
END

TO BOXES  :NUMBER  :X
REPEAT  :NUMBER [SETX  :X PD BOX PU MAKE "X  :X + 27] SETX - 130
END

TO BOXES2
PU SETY 0 PD
BOX
PU RT 90 FD 27 LT 90 PD
END
```

```
TO BOX
REPEAT 4 [FD 20 RT 90]
END

TO GETANSWER
CLEARTEXT
PRINT [HOW MANY BOXES?]
MAKE "ANSWER READNUMBER
TEST  :ANSWER =  :NUMBER +  :NUMBER2
IFTRUE PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE
IFFALSE CHECK
END

TO CHECK
CLEARTEXT
PRINT [COUNT THE BOXES AGAIN.]
PRINT [TYPE THE NUMBER.]
MAKE "ANSWER READNUMBER
TEST  :ANSWER =  :NUMBER +  :NUMBER 2
IFTRUE CLEARTEXT PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE
IFFALSE CLEARTEXT PRINT [TALK WITH YOUR TEACHER.] MAKE "DIDITRIGHT
  "FALSE
END

TO READNUMBER
MAKE "ANSWER REQUEST
TEST  :ANSWER = [ ]
IFTRUE PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER
TEST NOT NUMBER? FIRST  :ANSWER
IFTRUE PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER
IFFALSE OUTPUT FIRST  :ANSWER
END

TO READKEY
IF RC? OUTPUT READCHARACTER
OUTPUT ''
END

TO WAIT  :TIME
IF  :TIME = 0 STOP
WAIT  :TIME - 1
END
```

Subtracting

```
TO CHOOSENUMBERS
MAKE "NUMBER 1 + RANDOM 9
MAKE "NUMBER2 1 + RANDOM 9
IF  :NUMBER2  >  :NUMBER CHOOSENUMBERS
END

TO BOXES  :NUMBER  :X
REPEAT  :NUMBER [SETX  :X PD BOX PU MAKE "X  :X + 27]
END
```

```
TO SUBTRACTING
HT CS CLEARTEXT
CHOOSENUMBERS
PU SETXY - 130 60
BOXES  :NUMBER ( - 130 )
SUBTRACT
GETANSWER
IF NOT  :DIDITRIGHT STOP
PRINT [TRY ANOTHER ONE.] WAIT 90
SUBTRACTING
END

TO BOXES2
PU SETY 0 PD
BOX
PU RT 90 FD 27 LT 90 PD
END

TO BOX
REPEAT 4 [FD 20 RT 90]
END

TO SUBTRACT
PRINT ( SENTENCE [HIT THE "S" KEY]  :NUMBER2 [TIMES.] )
MAKE "KEEPTRACK 0
OPERATIONS
END

TO OPERATIONS
IF  :KEEPTRACK =  :NUMBER2 STOP
MAKE "OP READKEY
IF  :OP = "S ERASE.BOXES MAKE "KEEPTRACK  :KEEPTRACK + 1
OPERATIONS
END

TO ERASE.BOXES
PD PENCOLOR 0 BOX
PU LT 90 FD 27 RT 90 PD PENCOLOR 1
END

TO GETANSWER
CLEARTEXT
PRINT [HOW MANY BOXES?]
MAKE "ANSWER READNUMBER
TEST  :ANSWER =  :NUMBER -  :NUMBER2
IFTRUE PRINT [GOOD!] MAKE "DIDITRIGHT "TRUE]
IFFALSE CHECK
END
```

```
TO CHECK
CLEARTEXT PRINT [COUNT THE BOXES AGAIN.]
PRINT [TYPE THE NUMBER.]
MAKE "ANSWER READNUMBER
TEST  :ANSWER =  :NUMBER -  :NUMBER2
IFTRUE CLEARTEXT PRINT [GOOD!] MAKE "DIDITRIGHT "FALSE
IFFALSE CLEARTEXT PRINT [TALK WITH YOUR TEACHER.] MAKE "DIDITRIGHT
  "FALSE
END

TO READNUMBER
MAKE "ANSWER REQUEST
TEST  :ANSWER = [ ]
IFTRUE PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER
TEST NOT NUMBER? FIRST  :ANSWER
IFTRUE PRINT [PLEASE TYPE A NUMBER.] OUTPUT READNUMBER
IFFALSE OUTPUT FIRST  :ANSWER
END

TO READKEY
IF RC? OUTPUT READCHARACTER
OUTPUT ''
END

TO WAIT  :TIME
IF  :TIME = 0 STOP
WAIT  :TIME - 1
END
```

Bibliography

Abelson, Harold. TI Logo. New York: McGraw-Hill Book Company, 1984.

Bearden, Donna. 1, 2, 3 My Computer and Me. Reston: Reston Publishing Company. 1983.

Bearden, Donna. A Bit of Logo Magic. Reston: Reston Publishing Company, 1984.

Bearden, Donna; Martin, Kathleen; and Muller, Jim. The Turtle's Sourcebook. Reston: Reston Publishing Company, 1983.

Microquests, Martin-Bearden, Inc., Box 337, Grapevine, Texas 76051. Monthly newsletter for teachers.

National LOGO Exchange, Box 5431, Charlottesville, VA 22905. Monthly newsletter.

Pasternack, Marian and Silvey, Linda. Pattern Blocks Activities. Palo Alto, CA: Creative Publications, Inc. 1975.

Thornburg, David D. Discovering Apple Logo, An Invitation to the Art and Pattern of Nature. Reading, Mass.: Addison-Wesley Publishing Company, 1983.

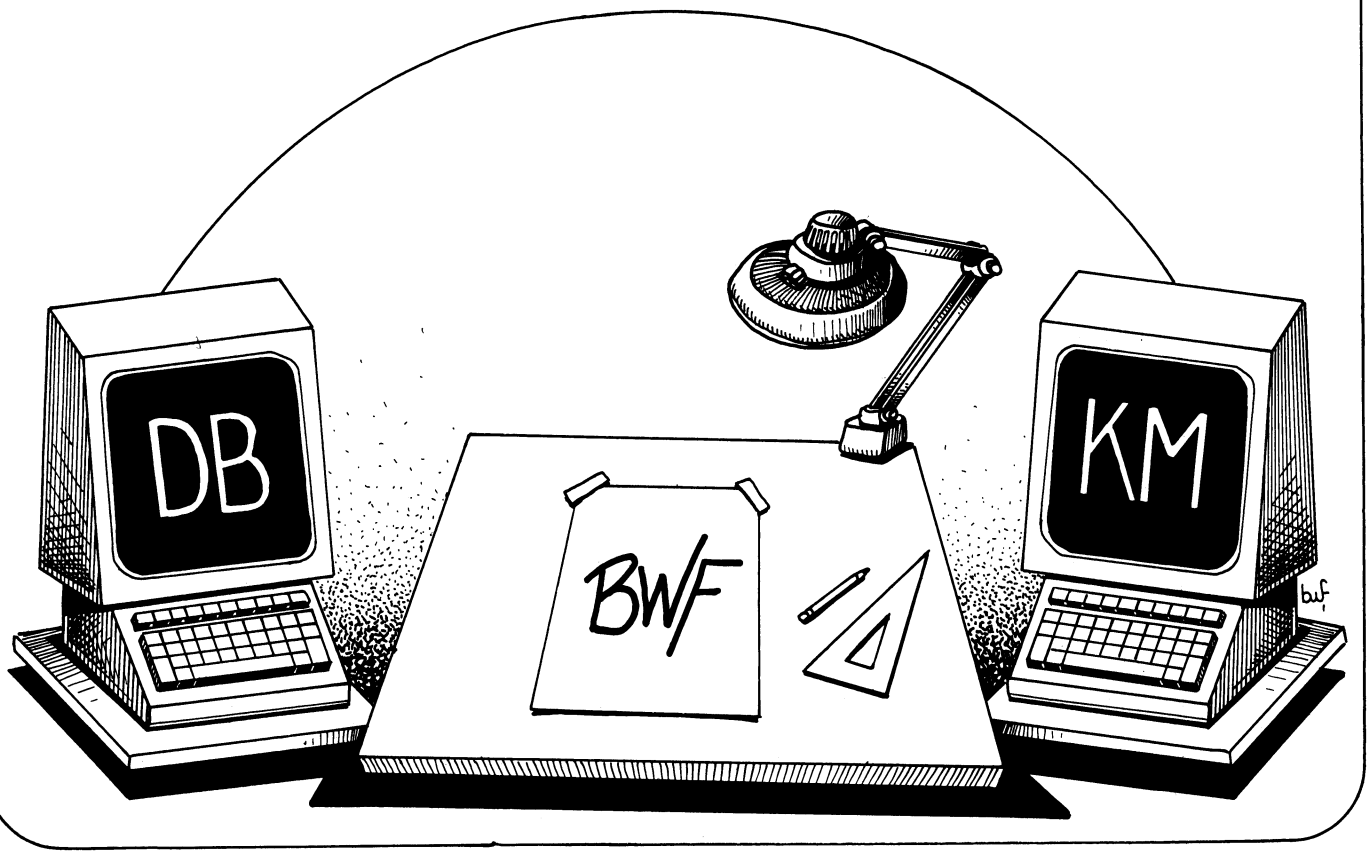
Thornburg, David D. Every Kids First Book of Robots and Computers. Greensboro, NC: Small Systems Services, Inc. 1982.

Watt, Daniel. Learning with LOGO. Highstown: Byte/McGraw-Hill, 1983.

DONNA BEARDEN is the author of several Logo books published by Reston including 1, 2, 3 My Computer and Me; A Bit of Logo Magic; and Atari Logo in the Classroom. She co-authored The Turtle's Sourcebook. Monica the Computer Mouse (Sybex), her first children's book, combines fantasy and fact to present information on how computers are used in our society. With Kathleen Martin, she publishes a monthly newsletter for teachers concerned with the creative use of computers in the classroom.

KATHLEEN MARTIN, Ph.D., is a professor of education at the University of Dallas. With extensive background in teaching and teacher education, she has been the recipient of four National Science Foundation grants to help elementary school teachers work with science and mathematics more creatively in the classroom. She co-authors a monthly column in The Computing Teacher and co-edits Microquests published by Martin-Bearden, Inc.

BRAD W. FOSTER is a free lance illustrator residing in Irving, Texas. This is the third Logo computer book he has illustrated for Reston Publishing Co., with more (he hopes) to follow. With author Donna Bearden, he illustrated his first children's book, Monica the Computer Mouse. He has also designed and illustrated a coloring album based on the classic Aesop's fables, from Troubador Press, and is presently working on a number of new children's books. His motto is "Don't bother me now, I've got a deadline to meet!"



Primarily LOGO

Donna Bearden • Kathleen Martin

At home or in the classroom, you can tell when Logo is being used. You'll know it by the enthusiasm of young learners as they pick up such concepts as spatial relationships, colors, and simple arithmetic operations as easily as 1-2-3. **Why?** Because Logo makes learning fun, not just for the gifted or for computer whiz-kids, but for nearly everyone.

Written for people with a bit of Logo experience, **PRIMARILY LOGO** shows teachers and parents of 5- to 9-year-olds how to use Logo and the computer as an effective educational combination.

This book covers pre-Logo activities, turtle graphics using full primitive commands, list processing, and model programs that help *you* come up with fresh ideas. The activities and exercises included in this book are designs for learning, *not* for learning programming. Teachers will find this especially useful as a means of integrating computer instruction into the traditional classroom.

Looking for more on Logo? Ask your local bookstore or teacher supply store to order some of these Reston books:

1-2-3 MY COMPUTER & ME
(Apple®/TI version by Donna Bearden and the Young People's Logo Association; Commodore 64 and Atari versions by Jim Muller and Y.P.L.A.)
A Logo funbook for kids that is written for beginners.

A BIT OF LOGO MAGIC
(Apple, TI and IBM® PC/PCjr versions by Donna Bearden)
Adventures for intermediate programmers taking a step beyond 1-2-3.

ATARI® LOGO IN THE CLASSROOM
(by Donna Bearden)
A teacher's manual that is really a guidebook full of useable ideas for the classroom.

THE TURTLE'S SOURCEBOOK
(for most versions of Logo by Donna Bearden, Kathleen Martin, Jim Muller, and the Young People's Logo Association)
A practical guide to learning and teaching with Logo.

MATHEMATICS AND LOGO
(by Kathleen Martin and Donna Bearden)
Covers a variety of math curriculum requirements the fun way—with Logo.

THE OTHER SIDE OF LOGO
(by Duanne McGriff, Andy Howard, and Donna Bearden)
A manual of list processing commands. There's a lot more to Logo than just turtle graphics.

A Reston Computer Group Book
Reston Publishing Company, Inc.
A Prentice-Hall Company
Reston, Virginia

0 21898 56771 0

0-8359-5677-6